

**AFRL-RI-RS-TR-2007-278**  
**Final Technical Report**  
**January 2008**



# **ENERGY AND POWER AWARE COMPUTING THROUGH MANAGEMENT OF COMPUTATIONAL ENTROPY**

**Georgia Institute of Technology**

**Sponsored by**  
**Defense Advanced Research Projects Agency**  
**DARPA Order No. M767**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**STINFO COPY**

**The views and conclusions contained in this document are those of the authors  
and should not be interpreted as necessarily representing the official policies,  
either expressed or implied, of the Defense Advanced Research Projects  
Agency or the U.S. Government.**

**AIR FORCE RESEARCH LABORATORY  
INFORMATION DIRECTORATE  
ROME RESEARCH SITE  
ROME, NEW YORK**

## NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2007-278 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/

WILMAR SIFRE  
Work Unit Manager

/s/

JAMES A. COLLINS  
Deputy Chief, Advanced Computing Division  
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

<b>REPORT DOCUMENTATION PAGE</b>				<i>Form Approved</i> <b>OMB No. 0704-0188</b>	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.</small>					
<b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b>					
<b>1. REPORT DATE (DD-MM-YYYY)</b> JAN 08		<b>2. REPORT TYPE</b> Final		<b>3. DATES COVERED (From - To)</b> May 02 – Jun 07	
<b>4. TITLE AND SUBTITLE</b>  ENERGY AND POWER AWARE COMPUTING THROUGH MANAGEMENT OF COMPUTATIONAL ENTROPY				<b>5a. CONTRACT NUMBER</b>	
				<b>5b. GRANT NUMBER</b> F30602-02-2-0124	
				<b>5c. PROGRAM ELEMENT NUMBER</b> 62301E	
<b>6. AUTHOR(S)</b>  Krishna V. Palem and Mark Richards				<b>5d. PROJECT NUMBER</b> EPAC	
				<b>5e. TASK NUMBER</b> 00	
				<b>5f. WORK UNIT NUMBER</b> 01	
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Georgia Institute of Technology 505 10 <sup>th</sup> St NW Atlanta GA 30332-0001				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b>  Defense Advanced Research Projects Agency      AFRL/RITB 3701 North Fairfax Dr.                                      525 Brooks Rd Arlington VA 22203—1714                                  Rome NY 13441-4505				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>	
				<b>11. SPONSORING/MONITORING AGENCY REPORT NUMBER</b> AFRL-RI-RS-TR-2007-278	
<b>12. DISTRIBUTION AVAILABILITY STATEMENT</b> APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 07-0709					
<b>13. SUPPLEMENTARY NOTES</b>					
<b>14. ABSTRACT</b> <p>With the ever increasing importance of computing in portable and other embedded settings, the power and energy consumed by computation has become an important, if not the central consideration driving research in computer engineering and science. Historically, the study of power and energy have roots in physics, notably in the classical field of thermodynamics. While thermodynamics was originally introduced and studied within the context of mechanical systems such as heat-engines, the underlying principles are equally applicable in the more modern context of processors fabricated from semiconductor materials. The innovations that we propose in this work aim to develop a mechanism for designing algorithms that are power (energy) aware by directly working with their thermodynamic properties, as opposed to the traditional approaches wherein algorithm design is concerned with measures such as their running time, and space consumed. Overall, our research scope was to innovate energy aware algorithms, novel semiconductor circuits, methodology and models for energy aware algorithm design by the application of thermodynamics and randomization toward algorithm and semiconductor design and analysis. As a result of this research work, we innovated probabilistic CMOS or PCMOS technology, as a promising approach to addressing the CMOS device scaling challenges and as a dramatic shift from previous work. Devices based on this technology, where noise is harnessed as a resource to implement CMOS devices exhibiting probabilistic behavior, are guaranteed to compute correctly with a probability p. Additionally, this work also characterized an explicit relationship between the probability p with which the CMOS switch computes correctly, and its associated physical attributes such as the energy consumed by each switching step across technology generations. We extended the above characterizations into PCMOS laws governing the behavior of such devices and switching. There laws were also validated across various technology generations via simulations as well as physical measurements of PCMOS inverters.</p>					
<b>15. SUBJECT TERMS</b> Power Aware, Entropy					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>  UL	<b>18. NUMBER OF PAGES</b>  60	<b>19a. NAME OF RESPONSIBLE PERSON</b> Wilmar Sifre
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			<b>19b. TELEPHONE NUMBER (Include area code)</b> N/A

# TABLE OF CONTENTS

<b>LIST OF FIGURES AND TABLES. ....</b>	<b>iii/iv</b>
<b>SUMMARY .....</b>	<b>1</b>
<b>1 INTRODUCTION .....</b>	<b>1</b>
1.1 Power Aware Computing Through Management of Computational Entropy	1
1.2 High Productivity Embedded Computing Technologies .....	2
1.3 Study in Reverse Engineering of Legacy Applications .....	2
1.4 Software Standards and Techniques for Cognitive and Commodity Computing Systems .....	3
<b>2 METHODOLOGY .....</b>	<b>4</b>
2.1 Power Aware Computing Through Management of Computational Entropy	4
2.1.1 A switch and switching. ....	4
2.1.2 Composing switches. ....	5
2.1.3 Inverter realization of a probabilistic switch in CMOS .....	6
2.1.4 Realizing low-energy PSoC architectures .....	10
2.1.5 PCMOS based architectures for error tolerant applications .....	13
2.2 High Productivity Embedded Computing Technologies. ....	14
2.2.1 Parametric machine model .....	15
2.3 Reverse Engineering of Legacy Applications .....	15
2.4 Software Standards and Techniques for Cognitive and Commodity Computing Systems .....	18
2.4.1 ACIP living framework forum task .....	18
2.4.2 GPU SVM task .....	18
<b>3 RESULTS .....</b>	<b>20</b>
3.1 Power Aware Computing Through Management of Computational Entropy	20
3.1.1 Analytical model and the three Laws of a PCMOS inverter . . . . .	20
3.1.2 Analysis and optimization of energy, performance, and probability of PCMOS circuits .....	25
3.1.3 Results and analysis for PSoC architectures. ....	25
3.1.4 BIVOS and DSP results .....	26

3.2 High Productivity Embedded Computing Technologies. ....	28
3.2.1 Survey of HECS . ....	28
3.2.2 Productivity challenges and a compiler centric approach . ....	40
3.3 Reverse Engineering of Legacy Applications . ....	40
3.4 Software Standards and Techniques for Cognitive and Commodity Computing Systems . ....	43
3.4.1 GPU SVM Task . ....	43
<b>4 CONCLUSIONS . ....</b>	<b>45</b>
4.1 Power Aware Computing Task . ....	45
4.2 HECS Task . ....	45
4.3 Reverse Engineering Task . ....	45
<b>REFERENCES . ....</b>	<b>46</b>
<b>LIST OF ACRONYMS . ....</b>	<b>50</b>

## LIST OF FIGURES

1	The inputs to a switch . . . . .	5
2	Symbols for inputs and outputs of a switch . . . . .	7
3	(a) Deterministic one bit switching functions (b) Their probabilistic counterparts with probability parameter (probability of correctness) $p$ . . . . .	6
4	Three types of switches . . . . .	6
5	A deterministic 2-canonical network resolving the AND function where the compute-switches $sw_3$ and $sw_4$ with input switching relationships $\langle x_{3,1}, w_{3,1} \rangle$ and $\langle x_{3,2}, w_{3,2} \rangle$ as well as wires $\langle y_{1,1}, w_{3,1} \rangle$ , $\langle z_{1,1}, x_{3,1} \rangle$ , $\langle y_{3,2}, w_{4,1} \rangle$ , $\langle z_{2,1}, x_{4,2} \rangle$ drive the <i>accepting</i> switch $sw_6$ and <i>rejecting</i> switch $sw_5$ . . . . .	7
6	(a) An inverter coupled with thermal noise at its output (b) An idealized inverter transfer curve (c) Probabilistic behavior determined by thermal noise coupled at the output of the inverter. . . . .	8
7	(a) Input voltage of a deterministic CMOS inverter (b) Output voltage of a deterministic CMOS inverter (c) Output voltage of a probabilistic CMOS inverter with probability parameter $p = 0.87$ for the same input . . . . .	9
8	A “SoC-like” architecture for realizing low-energy computing architectures using PCMOS. . . . .	10
9	The four possible realizations of an application using an SoC platform wherein (a) a deterministic application is executed entirely on the host, (b) a probabilistic application is executed entirely on the host, using an emulation based on pseudo-random bits generated using software, (c) the emulation is realized using a custom CMOS co-processor, and (d) a functionally identical PCMOS co-processor is used to realize the probabilistic components of the application. . . . .	12
10	The host and co-processor architecture and power modeling. . . . .	13
11	Typical design flow for (a) embedded computing systems, and (b) heterogeneous embedded computing systems . . . . .	17
12	Example for state-of-the-art tools by Semantic Designs, Inc. . . . .	19
13	PCMOS chip fabricated in AMI 0.5 $\mu$ m technology . . . . .	21
14	PCMOS chip fabricated in TSMC 0.25 $\mu$ m technology . . . . .	22
15	Law 1 validated over 0.5 $\mu$ m and 0.25 $\mu$ m processes . . . . .	23
16	Law 2 validated over 0.5 $\mu$ m and 0.25 $\mu$ m processes . . . . .	23
17	Validation of invariance law of PCMOS through simulations and measurements. The invariant behavior of $p$ -NSR relationship is shown for four technology generations, 90nm, 65nm, 0.25 $\mu$ m and 0.5 $\mu$ m, based on HSpice simulations as well as physical measurements of PCMOS inverters fabricated using 0.25 $\mu$ m and 0.5 $\mu$ m processes . . . . .	24

18	(a) Conventional digital design with very high (nominal) voltage levels. (b) Our probabilistic, BIVOS approach with significantly lower energy consumed and leasing to minimal degradation of the quality of the image. (c) Conventional voltage scaling that achieves the same level of energy savings as (b) but with significantly lower image quality. . . . .	27
19	A canonical heterogeneous computing system. . . . .	28
20	Model for heterogeneous embedded system architectures . . . . .	29
21	Attributes of a programmable tile . . . . .	30
22	Relation between programmable tile attributes and platform attributes . . . . .	30
23	Comparing platforms with their quantifiable attributes . . . . .	31
24	Key costs of HECS design flow . . . . .	33
25	Tools and their activities (this example relates languages to activities) . . . . .	34
26	Application domains . . . . .	35
27	Higher computational complexity correlates with higher platform power . . . . .	38
28	Higher computational complexity inversely correlates with platform mobility . . . . .	39
29	Hurdles to productivity . . . . .	41
30	A compiler centric approach to massively improved productivity . . . . .	42

## LIST OF TABLES

1	Application level maximum and minimum EPP gains of PCMOS over the baseline implementation where the implementation <i>I</i> has a StrongARM SA-1100 host and a PCMOS based co-processor . . . . .	26
2	Comparing the quality of the output images achieved through our novel BIVOS based PCMOS to those achieved through conventional voltage scaling and the corresponding energy savings of both approaches when compared to nominal operation at full scale $V_{dd}$ . . . . .	27
3	Constructive plan summary for implementing our proposed approach . . . . .	41

## SUMMARY

Sustained device scaling into the nanometer regime faces several hurdles. Manufacturing difficulties yield devices with parameter variations and since these devices are likely to operate close to the thermal limit, they are susceptible to perturbations due to noise. Clearly, current circuit design methodologies are inadequate to design robust circuits in the presence of these perturbations, since they depend on the devices with deterministic behavior. To design robust circuits and architecture in the presence of this (inevitable) statistical behavior at the device level, it has been speculated that a shift in the design paradigm—from the current day deterministic designs to statistical or probabilistic designs of the future—would be necessary.

Motivated by this necessity and in a departure from conventional approaches to modeling and characterizing the behavior of CMOS devices and switches deterministically, we provided an explicit probabilistic characterization of the behavior of computing switches through a novel probabilistic CMOS (PCMO) technology. In our research work, we have addressed this issue of probabilistic design at several levels, from foundational models to devices, circuits, and practical system-on-a-chip architectures which leverage PCMO technology for applications from the cognitive and embedded domains. In the architectural aspect of our work and in a first-of-a-kind demonstration, we were able to establish that probabilistic algorithms can simultaneously yield improvements not only in terms of the energy consumed but also in terms of the performance (or running-time) quantified through the energy-performance product (EPP) metric.

Additionally, this work also characterized an explicit relationship between the probability  $p$  with which the CMOS switch computes correctly, and its associated physical attributes such as the energy consumed by each switching step across technology generations. These characterizations were extended into PCMO laws governing the behavior of such devices and switching. These laws were also validated across various technology generations via simulations as well as physical measurements of PCMO inverters. On the other hand, we also performed an analysis and optimization of energy, performance, and probability of PCMO circuits. We showed the design trade-offs between energy, performance, and  $p$  of PCMO gates using analytical models of energy, propagation delay, and  $p$ .

When applied to the digital signal processing (DSP) domain, the resulting error in the output of a probabilistic arithmetic primitive, such as an adder for example, manifests as degradation in the signal-to-noise ratio (SNR). In return for this degradation that is enabled by our probabilistic arithmetic primitives—degradation visually indistinguishable from an



image reconstructed using conventional deterministic approaches—significant energy savings and performance gains are shown to be possible per unit of SNR degradation. These savings stem from a novel method of voltage scaling, which we refer to as biased voltage scaling (or BIVOS), that is one of the major technical innovations on which our probabilistic designs are based.

Historically, probability and statistics have played a central role in characterizing the behavior of physical systems, and in enabling significant technological advances. For example, starting with the foundations laid by historical figures such as Clausius [12] and Carnot [5], classical thermodynamics was revolutionized by giants such as Maxwell [28], Boltzmann [4] and Gibbs [18] through the introduction of probability into considerations involving the irreversibility implied by the celebrated “second law.” Moving a century forward in history and considering the information revolution matching if not surpassing the industrial revolution associated with thermodynamics, again, deterministic considerations prevailed. Thus, the profound foundations due to Turing, and its pragmatic realization characterized by von Neumann [47] and engineered by Eckert and Mauchly [21], resulted in computers whose behavior is error free, at least in intention. This changed in a dramatic turn of events, best described by Schwartz [40] who commented on the (then) revolutionary idea of “randomized” or probabilistic algorithms: “The startling success of the Rabin-Strassen-Solovay (see Rabin [38]) algorithm, together with the intriguing foundational possibility that axioms of randomness may constitute a useful fundamental source of mathematical truth independent of, but supplementary to, the standard axiomatic structure of mathematics (see Chaitin and Schwartz [6]), suggests that probabilistic algorithms ought to be sought vigorously.”

While these probabilistic algorithms have influenced the field of computing in the broadest sense notably through methods for computer security based on cryptography, the underlying “hardware” that serves as a platform for computing has remained “deterministic”. However, as the feature sizes of CMOS transistors—the atomic constructs in modern computing—approach the low nanometer range, “probabilistic” behaviors are a serious and potentially unavoidable consideration. Thus, characterizing the behavior of transistors in a probabilistic setting is crucial to understanding and enabling the design of computing platforms in the future. We believe that the laws of PCMOS that we derived through this work constitute an important step in this direction and provide a practical foundation for realizing highly efficient probabilistic architectures. In this context, through PCMOS based architectural designs, curiously, noise which is viewed as an impediment within the context of technology scaling can in fact be turned into an asset. By managing it carefully, it can be used to derive significant savings in the energy-performance sense.

# CHAPTER 1

## INTRODUCTION

This report presents the research work performed under the following four main tasks entitled:

- Power Aware Computing Through Management of Computational Entropy
- High Productivity Embedded Computing Technologies
- Study in Reverse Engineering of Legacy Applications
- Software Standards and Techniques for Cognitive and Commodity Computing Systems

We will now introduce the scope of each of these tasks in the subsequent sections.

### ***1.1 Power Aware Computing Through Management of Computational Entropy***

In this research work, we innovated models of computing for energy-aware algorithm design and analysis, for the first time, based on the following thesis: *The computational technique referred to as randomization yielding probabilistic algorithms, now ubiquitous to the mathematical theory of probabilistic algorithm design and analysis, when interpreted as a physical phenomenon through classical statistical thermodynamics, yields to energy savings that decrease with the probability  $p$  with which each primitive computational step is guaranteed to be correct (or, equivalently, increase with the probability of error,  $(1 - p)$ ).*

As a result of this research work, we innovated probabilistic CMOS or PCMOS technology, as a promising approach to addressing the CMOS device scaling challenges and as a dramatic shift from previous work. Devices based on this technology, where noise is harnessed as a resource to implement CMOS devices exhibiting probabilistic behavior, are guaranteed to compute correctly with a probability  $p$ . Here,  $p$  is a design parameter; and by design, the devices are expected to compute incorrectly with a probability  $(1 - p)$ . The foundations of PCMOS technology are rooted in physics of computation, algorithms and information theory.

Earlier, using techniques derived from physics of computation and information theory, we showed that the thermodynamic cost of computing a bit of information is directly related to its probability  $p$  of being correct [32]. This proof uses purely entropic arguments derived from the second law of thermodynamics [33]. Further, using an abstract model of computation, the RABRAM [32], and using the example of the distinct vector problem [32], we demonstrated that such energy savings at the switching level, can be harnessed at the application level to construct a probabilistic algorithm (the value amplification algorithm [32]) that is more (energy) efficient than the best possible deterministic algorithm.

While the work mentioned above demonstrates the energy efficacy and utility of probabilistic behavior in abstract computational models, the foundational principles were also extended into the CMOS domain through a systematic characterization of a PCMOS device [10, 11, 25, 26, 27]. This characterization involves the study of the relationship

between the probability  $p$  of reliable operation, the noise level and the switching energy. Further, this relationship (captured by the two PC MOS laws) was verified through simulations as well as measurements of PC MOS devices fabricated in TSMC 0.25 $\mu$ m and AMI 0.5 $\mu$ m processes. Through analytical modeling, simulations, and measurements of fabricated PC MOS switches, we showed that, while devices based on PC MOS technology exhibit probabilistic behavior due to low-voltage operation and noise susceptibility, they achieve extreme energy savings in return. Further, we demonstrated the utility of PC MOS technology to computing platforms by studying application specific architectures that can not only harness PC MOS technology to implement real world applications, but also can lead to extremely efficient implementations—both in terms of energy measured in Joules, as well as performance (running time) measured in seconds, simultaneously captured by the *energy x performance* metric—when compared to conventional CMOS based designs. The device that we study is a PC MOS based inverter, ubiquitous to digital design and a key building block used in probabilistic applications. In the context of a probabilistic system-on-a-chip (PSOC), we demonstrated the utility and benefits of PC MOS based architectures in implementing probabilistic applications such as pattern recognition, optimization, classification, patient monitoring, and Windows printer troubleshooting through randomized neural networks [16, 15], probabilistic cellular automata [14], hyper-encryption [13], and Bayesian inferencing [39, 37, 3] (see our publications [7, 9]).

Besides the aforementioned probabilistic applications, we also showed the utility and benefits of PC MOS based probabilistic computing in the context of error-tolerant applications, such as synthetic aperture radar (SAR) imaging and video decoding (see [17, 1]). Here, through the use of probabilistic arithmetic, device-level bit errors—which translate into image quality measured through SNR at the application-level—can be mitigated and traded for energy savings with minimal impact on application quality [17].

We will describe all of the work done introduced above in the following sections of the report: see Chapter 2.1 and Chapter 3.1.

## **1.2 *High Productivity Embedded Computing Technologies***

This work studied key challenges facing development and deployment of heterogeneous embedded systems. In particular, the program addresses productivity challenges that preclude realization of the anticipated advantages of heterogeneous embedded computing systems (HECS). Our effort provides a (parametric) system model to characterize a mix of custom processors, polymorphous architectures, and domain accelerator extensions. Architecture space modeling, design space exploration and productivity enhancing optimizations for HECS are the key steps involved in this work.

We will describe the work done under this task in the following sections of the report: see Chapter 2.2 and Chapter 3.2.

## **1.3 *Study in Reverse Engineering of Legacy Applications***

In parallel with the research work on HECS (see Section 1.2), we performed a proof-of-concept study in reverse engineering of legacy applications. Specifically, legacy high-level languages tend to be in formats and forms that are not readily amenable to human interpretation, redesign and deployment. This task involves a survey on what and where

such legacy applications are, the hurdles of legacy code, the state-of-the-art approaches to deal with legacy code, as well as a constructive plan for addressing hurdles through cognitive and probabilistic techniques.

We will describe the work done under this task in the following sections of the report: see Chapter 2.3 and Chapter 3.3.

#### ***1.4 Software Standards and Techniques for Cognitive and Commodity Computing Systems***

From October 2004 through August 2006, research was conducted in software standards and techniques for heterogeneous computing systems. This research focused on two sub-tasks:

(1) Assessing the need and planning for a potential “Living Framework Forum” (LFF) software architecture task for the DARPA Architectures for Cognitive Information Processing (ACIP) program.

(2) Development, testing, and evaluation of an implementation of the PCA SVM API (Polymorphous Computing Architectures (PCA) Stream Virtual Machine (SVM) application programming interface (API)) for commodity graphical processing units (GPUs).

We will describe the work done under each of these sub-tasks in the following sections of the report: see Chapter 2.4 and Chapter 3.4.

## CHAPTER 2

### METHODOLOGY

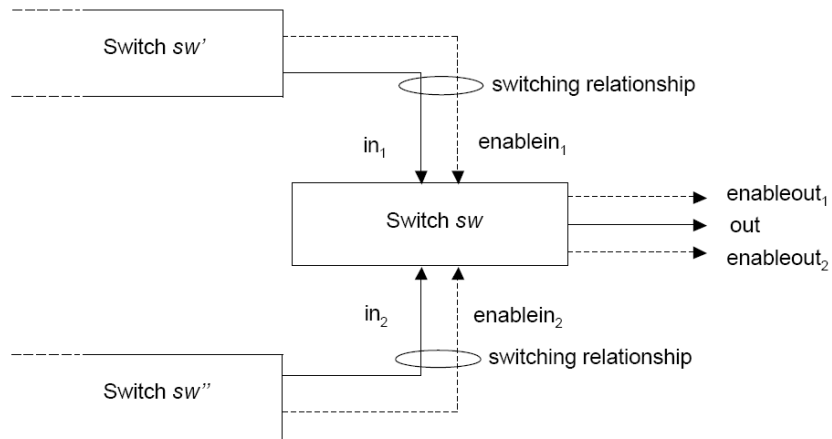
#### 2.1 *Power Aware Computing Through Management of Computational Entropy*

Our work on energy aware computing through probabilistic switching took a different view from existing approaches by interpreting probability to be a byproduct of a physical phenomenon, ubiquitous to nature. In contrast to the universal view, wherein noise is viewed as an impediment, our approach took the diametrically opposed view of viewing it as a resource as an aid to achieving low-energy probabilistic devices and computing. Thus, most devices based on CMOS or other physical media are, by their very nature, unstable, noisy or, from our perspective, inherently probabilistic. In the sequel, we will first present our switch-based gate constructions and methodology developed for building energy aware networks for computing, using probabilistic bits (PBITS).

##### 2.1.1 A switch and switching

Starting with an informal introduction, as shown in Figure 1, each switch  $sw$  has (up to) two alternate choices for “input values” as well as “enabling signals”. Each input value and enabling signal of  $sw$  is in turn the output of a distinct switch (from the set of all switches,  $\mathbf{SW}$ ),  $sw'$  and  $sw''$  in the example. The outputs of switch  $sw'$  are identified with the input value  $in_1$  and the input enabling signal  $enablein_1$ , whereas the outputs of  $sw''$  are identified with  $in_2$  and  $enablein_2$ . Any switch  $sw$  in turn has two possible (mutually exclusive) enabling signals as output denoted by  $enableout_1$  and  $enableout_2$ , as well as a single output value  $out$ .

During the entire lifetime of a switch  $sw$ , each of its enabling signals  $enablein_i$ ,  $i \in \{1, 2\}$ , is “associated with” exactly one  $in_j$ ,  $j \in \{1, 2\}$ . Subsequently, these associations will be formalized as “switching relationships”. As shown in Figure 1,  $enablein_1$  is associated with  $in_1$  and similarly,  $enablein_2$  is associated with  $in_2$ ; in general, all four possible associations



**Figure 1:** The inputs to a switch

Signal	Symbol
enablein	w
in	x
enableout	y
out	z

**Figure 2:** Symbols for inputs and outputs of a switch

are allowed. In any legal *switching* of  $sw$ , *exactly one* of its two enabling signals must be “active”, indicated by associating the value 1 with it. Finally, in this example, switch  $sw$  produces an output value as a function of  $in_1$  whenever  $enablein_1$  is active, whereas it produces an output value as a function of  $in_2$  whenever  $enablein_2$  is active where each switch  $sw$  realizes one of the four possible 1-bit functions (i.e., identity, complement, 0 constant, and 1 constant).

To further understand switching, let us suppose that  $enablein_1 = 1$ . Recall from Figure 1 that the association or switching relationship between  $enablein_1$  and  $in_1$  implies that whenever  $enablein_1 = 1$ , out is determined by  $in_1$ . In this case,  $sw$  now switches and produces an output using function  $f$ .

Given a switch  $sw$  with an associated function  $f$ , a *switching* (step) is defined as follows:

1.  $z = \Gamma$  and  $y_1 = y_2 = 0$  whenever both its input enabling signals  $w_1$  and  $w_2$  have an identical value.
2. Whenever exactly one input enabling signal say  $w_i = 1$  (and  $w_{i'} = 0$  for  $i \neq i'$ )  $\langle w_i, x_j \rangle$  is a valid switching relationship,
  - (a) if  $x_j = 0$  then  $z = f(0)$ ,  $y_1 = z$  and  $y_2 = \bar{z}$
  - (b) if  $x_j = 1$  then  $z = f(1)$ ,  $y_1 = z$  and  $y_2 = \bar{z}$

Let  $f(x) = z$  be the *deterministic* switching realized by  $sw$ . A *probabilistic switching* with a probability parameter  $p \geq \frac{1}{2}$  is defined to be  $f(x) = z$  with a probability  $p$  and  $f(x) = \bar{z}$  with probability  $(1 - p)$ .

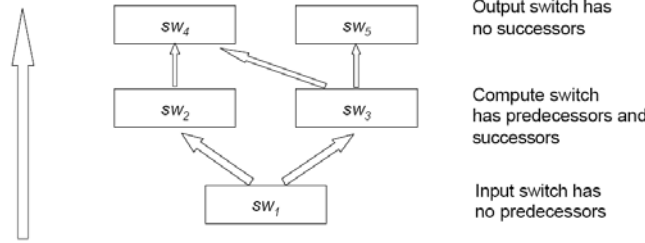
As illustrated in Figure 3, the four deterministic one bit switching functions (Figure 3(a)) have a probabilistic counterpart (Figure 3(b)) with an explicit probability parameter (probability of correctness)  $p$ . Of these, the two constant functions are trivial and the others are nontrivial. We consider an abstract probabilistic switch  $sw$  to be the one which realizes one of these four probabilistic switching functions. Such elementary probabilistic switches are composed to realize primitive Boolean functions, such as AND, OR, NOT functions.

### 2.1.2 Composing switches

To develop structures meant to realize entire computations, we identify three types of switches: INPUT-SWITCH, OUTPUT-SWITCH and COMPUTE-SWITCH as shown in Figure 4. An INPUT-SWITCH  $sw_1$  has no predecessors and drives at least one switch of type COMPUTE-SWITCH or of type OUTPUT-SWITCH. A switch such as  $sw_2$  in our example, which is a COMPUTE-SWITCH, is driven by a switch  $sw_1$  which is either an INPUT-SWITCH or a COMPUTE-SWITCH. A COMPUTE-SWITCH can in turn drive one or more switches that are

<table><tr><th>Input</th><th>output</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table> <p>Identity Function</p>	Input	output	0	0	1	1	<table><tr><th>Input</th><th>output</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table> <p>Complement Function</p>	Input	output	0	1	1	0	<table><tr><th>Input</th><th colspan="2">output</th></tr><tr><td>0</td><td>0 (p)</td><td>1 (1-p)</td></tr><tr><td>1</td><td>1 (p)</td><td>0 (1-p)</td></tr></table> <p>Identity Function</p>	Input	output		0	0 (p)	1 (1-p)	1	1 (p)	0 (1-p)	<table><tr><th>Input</th><th colspan="2">output</th></tr><tr><td>0</td><td>1 (p)</td><td>0 (1-p)</td></tr><tr><td>1</td><td>0 (p)</td><td>1 (1-p)</td></tr></table> <p>Complement Function</p>	Input	output		0	1 (p)	0 (1-p)	1	0 (p)	1 (1-p)
Input	output																																
0	0																																
1	1																																
Input	output																																
0	1																																
1	0																																
Input	output																																
0	0 (p)	1 (1-p)																															
1	1 (p)	0 (1-p)																															
Input	output																																
0	1 (p)	0 (1-p)																															
1	0 (p)	1 (1-p)																															
<table><tr><th>Input</th><th>output</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td></tr></table> <p>Constant Function</p>	Input	output	0	0	1	0	<table><tr><th>Input</th><th>output</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td></tr></table> <p>Constant Function</p>	Input	output	0	1	1	1	<table><tr><th>Input</th><th colspan="2">output</th></tr><tr><td>0</td><td>0 (p)</td><td>1 (1-p)</td></tr><tr><td>1</td><td>0 (p)</td><td>1 (1-p)</td></tr></table> <p>Constant Function</p>	Input	output		0	0 (p)	1 (1-p)	1	0 (p)	1 (1-p)	<table><tr><th>Input</th><th colspan="2">output</th></tr><tr><td>0</td><td>1 (p)</td><td>0 (1-p)</td></tr><tr><td>1</td><td>1 (p)</td><td>0 (1-p)</td></tr></table> <p>Constant Function</p>	Input	output		0	1 (p)	0 (1-p)	1	1 (p)	0 (1-p)
Input	output																																
0	0																																
1	0																																
Input	output																																
0	1																																
1	1																																
Input	output																																
0	0 (p)	1 (1-p)																															
1	0 (p)	1 (1-p)																															
Input	output																																
0	1 (p)	0 (1-p)																															
1	1 (p)	0 (1-p)																															

**Figure 3:** (a) Deterministic one bit switching functions (b) Their probabilistic counterparts with probability parameter (probability of correctness)  $p$



**Figure 4:** Three types of switches

either of type COMPUTE-SWITCH or of a type OUTPUT-SWITCH. In our example, COMPUTE-SWITCH  $sw_2$  drives  $sw_4$  which is an OUTPUT-SWITCH; an output switch has no successors and is driven by at least one switch of type INPUT-SWITCH or COMPUTE-SWITCH.

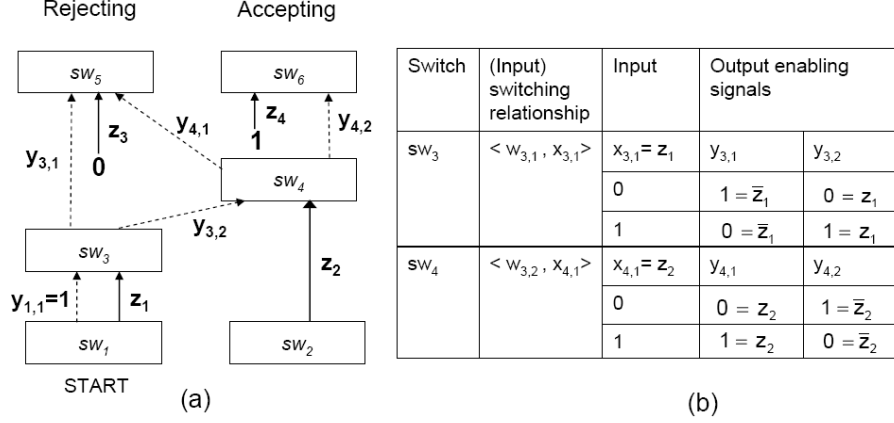
A *network* of switches is a connected directed acyclic graph  $\mathcal{N} = (\mathbf{SW}, \mathbf{WIRES})$  such that the vertices are switches, the edges are wires and the switches that are of the type OUTPUT-SWITCH as well as those of type COMPUTE-SWITCH are all well-connected. As an example, a network computing the logical AND function is sketched in Figure 5(a). Whereas switches  $sw_1$ ,  $sw_2$  are input switches, switches  $sw_3$ ,  $sw_4$  are compute switches (implementing the complement function) whereas switches  $sw_5$  and  $sw_6$  are output switches;  $sw_5$  is the rejecting switch and  $sw_6$  is the accepting switch. Every switch in this network, unless it is an INPUT-SWITCH, is well connected, and the network is directed and acyclic. In Figure 5(b), we show the crucial relationships between the input values to  $sw_3$  and  $sw_4$ , and their output and enabling signals in a “truth-table-like” structure.

Until now, we have briefly introduced our methodology to understand switching and networks. The details of our methodology can be found in our publication [33]. We have also included the concept of probabilistic switching with plausible device realizations to save energy in our patent application [35]. In [35], we showed an introverted switch in probabilistic designs as a basis for energy savings and for dealing with the increasing and significant challenge posed by static dissipation due to leakage.

The subsequent sections present the details of PCMOs technology, where our foundational principles outlined above are extended into the CMOS domain through systematic characterization of PCMOs devices and architectures derived from such devices.

### 2.1.3 Inverter realization of a probabilistic switch in CMOS

In this section, we show our characterization of a switch rendered probabilistic due to thermal noise. We first explain a CMOS inverter realization of a probabilistic switch. We



**Figure 5:** A deterministic 2-canonical network resolving the AND function where the compute-switches  $sw_3$  and  $sw_4$  with input switching relationships  $\langle x_{3,1}, w_{3,1} \rangle$  and  $\langle x_{3,2}, w_{3,2} \rangle$  as well as wires  $\langle y_{1,1}, w_{3,1} \rangle$ ,  $\langle z_1, x_{3,1} \rangle$ ,  $\langle y_{3,2}, w_{4,1} \rangle$ ,  $\langle z_2, x_{4,2} \rangle$  drive the *accepting* switch  $sw_6$  and *rejecting* switch  $sw_5$

then develop an analytical model and the associated laws of PCMOs technology, crucial to understanding the probabilistic behavior of a PCMOs switch, and then validate the model via HSpice simulations.

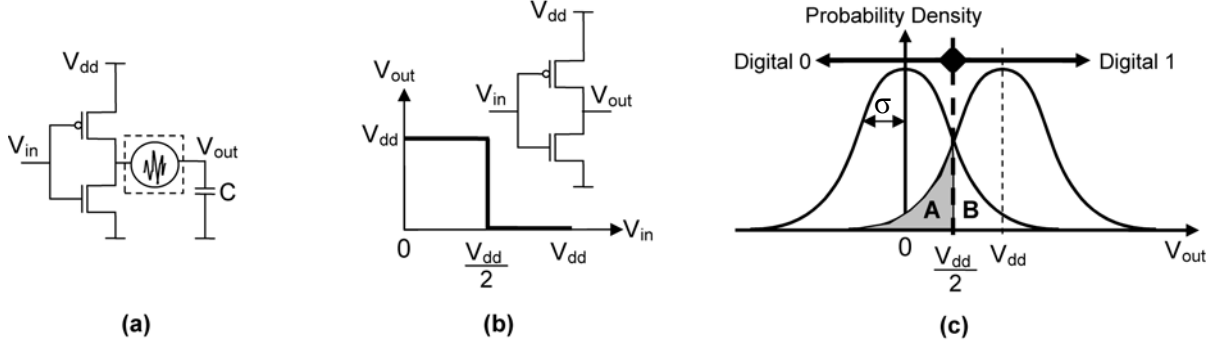
A CMOS inverter is a digital gate that executes the *inversion* function with one input and one output. The switching in this case is the invocation of the inversion function, which, in the context of the CMOS inverter, corresponds to the flow of the switching current through the output capacitance of the inverter. For a deterministic inverter,  $Y(t_2) = \overline{X(t_1)}$ , where  $X$  and  $Y$  denote the binary values of the input and the output of the inverter, respectively,  $t_2$  denotes the point in time when the switching ends and  $t_1$  denotes the point in time when the switching starts. For a probabilistic inverter, on the other hand,

$$Y(t_2) = \begin{cases} \overline{X(t_1)} & \text{with probability } p \\ X(t_1) & \text{with probability } (1 - p) \end{cases}, \quad (25.1)$$

wherein  $p$  denotes the probability of correctness, such that  $1/2 < p < 1$ . In Equation (1), the probability  $p$  results from the noise coupled to the CMOS inverter.

Following Stein [43], noise can be modeled as being coupled to the output of the inverter as shown in Figure 6(a). To understand how noise induces probabilistic behavior, we first consider the transfer characteristics of an ideal inverter shown in Figure 6(b). As shown in the figure, a single *switching* step of an ideal deterministic inverter is instantaneous and occurs at a value of  $V_{dd}/2$ . The binary values of 0 and 1 correspond to a (measured) output voltage in the interval  $(-\infty, V_{dd}/2)$  and  $[V_{dd}/2, +\infty)$ , respectively. However, noise that is present at the output node interacts with the voltage values representing a deterministic 0 or 1 corresponding to the output signal. Noise which is characterized by a Gaussian distribution with a standard variation  $\sigma$  is superimposed on the output signal, and therefore, it destabilizes the output of the inverter, causing incorrect switchings. To characterize the erroneous behavior and establish a relationship between noise magnitude, signal magnitude and the probability of correctness,  $p$ , we refer to the digital 0 and 1 regions shown in Figure 6(c). This figure corresponds to the case when the inverter is coupled with thermal noise at its output (see Figure 6(a)). The thermal noise has a Gaussian distribution with a standard deviation of  $\sigma$ , also referred to as the RMS value of noise. Here,



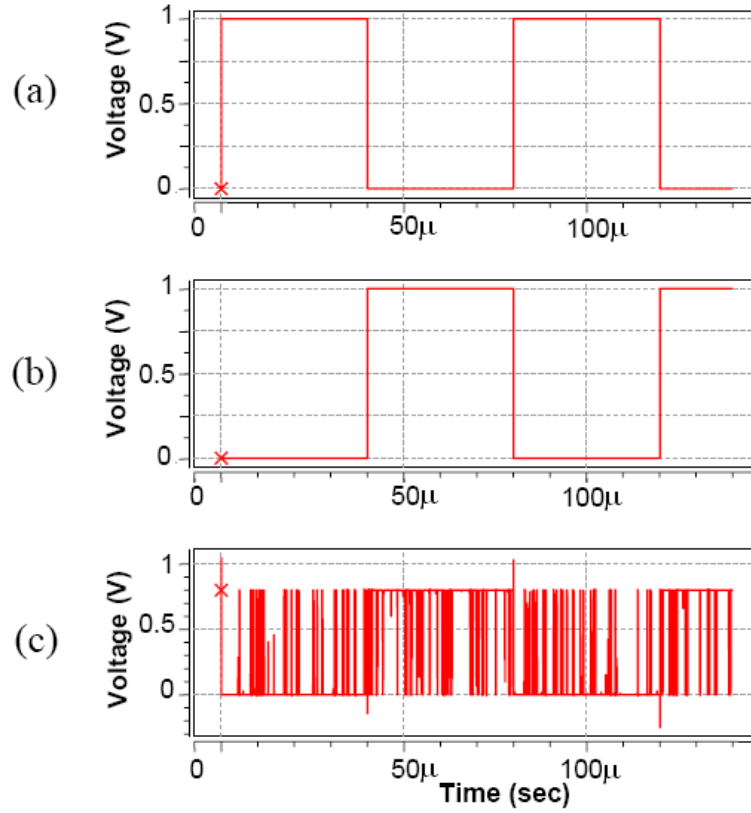


**Figure 6:** (a) An inverter coupled with thermal noise at its output (b) An idealized inverter transfer curve (c) Probabilistic behavior determined by thermal noise coupled at the output of the inverter

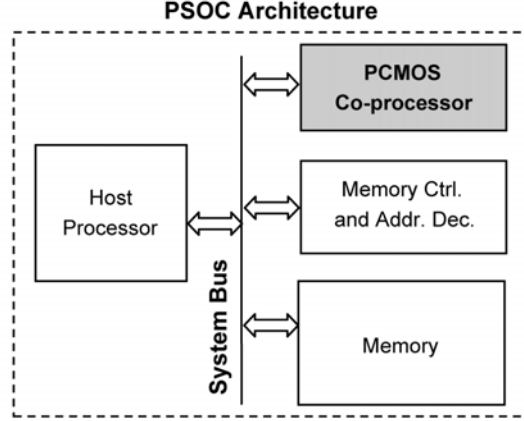
the curve on the left (right) has a mean of  $0V$  ( $1V$ ) and it corresponds to the case when the input  $V_{in}$  is  $1$  ( $0$ ). Thus, with an output value of  $1$  for example—measured to be  $V_{dd}$  in the ideal case—the instantaneous value will be determined by the noise distribution. Thus, while the output value ought to be  $V_{dd}$ , because of additive noise, it can easily be in the interval  $(-\infty, V_{dd}/2)$  inherently yielding a value of  $0$ . In Figure 6(c), the probability of error corresponding to an output value of  $1$  being erroneously treated as  $0$  is equal to the area **A**, and that of  $0$  being erroneously treated to be  $1$  corresponds to the area **B**. Note that, from symmetry,  $A=B = (1 - p)$ , where  $p$  is probability of being correct (or, equivalently,  $(1 - p)$  is probability of error) in both cases.

To understand the behavior of a probabilistic inverter, we compare the output waveforms of a deterministic inverter with that of a probabilistic inverter for the same input signal. The input signal waveform is shown in Figure 7(a). The corresponding output voltage waveform of a deterministic inverter is shown in Figure 7(b), and the corresponding output of a probabilistic inverter is shown in Figure 7(c). In Figure 7(c), the inverter is coupled with thermal noise at its input and it is designed to switch correctly with a probability parameter  $p = 0.87$ . Because of the noise, the output voltage of the inverter undergoes transitions to binary  $0$ , while it should be at binary  $1$ , and vice versa. Such a probabilistic inverter is of great value and we will present the utility of such a switch in Section 2.1.4, wherein the benefits in terms of energy savings and performance improvements are studied via an example application with probabilistic workload.

Note that in Figure 7, the output voltage level for a deterministic inverter is higher ( $1V$ ) than it is for a probabilistic inverter ( $0.8V$ ). This is because, the probabilistic behavior for the inverter is realized through varying two parameters (1) the noise amount coupled on the inverter characterized as its RMS value in Volts, and (2) supply voltage  $V_{dd}$  of the inverter, and in Figure 7(c), the supply voltage value of  $0.8V$  corresponds to a probability value  $p = 0.87$  with a noise RMS value of  $0.4V$ . The details of the effects of the two parameters, the amount of noise and the supply voltage, will constitute the two laws and are detailed in the following sections. Briefly, since the probability  $p$  results due to noise destabilizing the inverter shown in Figure 6(a), the probability parameter  $p$  is decreased either by increasing the noise (RMS) magnitude, or by decreasing the operating supply voltage of the inverter,  $V_{dd}$ . As a result, incorrect switchings occur at the output of the inverter as shown in Figure 7(c).



**Figure 7:** (a) Input voltage of a deterministic CMOS inverter (b) Output voltage of a deterministic CMOS inverter (c) Output voltage of a probabilistic CMOS inverter with probability parameter  $p = 0.87$  for the same input



**Figure 8:** A “SoC-like” architecture for realizing low-energy computing architectures using PCMOs

With this as background, we will present the result of the analytical model of the PCMOs inverter and then the associated laws deduced from this model in Chapter 3.1.

Whereas the methodology discussed above was based on noise being freely available, due to its limited availability in reality, the practical realization of a PCMOs inverter also needs amplifier circuitry, which incurs additional energy and time costs. In evaluating the architecture level benefits of PCMOs switches for the currently available technologies, we also consider the cost of amplification. The particular low-energy subthreshold amplifier’s structure and its attributes can be found in the work of Cheemalavagu, Korkmaz, Palem, Akgul, and Chakrapani [11]. In the next section, we use these PCMOs switches as a building block, and demonstrate the architecture level gains of PCMOs based system-on-a-chip implementations in the context of an application with a probabilistic workload.

#### 2.1.4 Realizing low-energy PSoC architectures

In order to better understand the energy and performance benefits of PCMOs technology that can be derived at the architectural level, we propose a system-on-a-chip (SoC) architecture as shown in Figure 8. PCMOs is utilized in the design of an *application-specific* co-processor, and the probabilistic content of the computation is executed on this co-processor. Thus, we envision early yet significant adoption of PCMOs to be application specific, and evolving to a context that is domain specific; for details about the concept of a SoC and further details about custom-fit processors, please see Lyonnard et al. [29], Tensilica [49] and Wang et al. [48]. As shown in the figure, a low-energy host processor is used to compute the deterministic components of the application. A typical host processor will be a StrongARM [45], a MIPS [42] or an equivalent low-energy embedded processor, coupled to the co-processor through the system bus. Thus, the communication between the host and the co-processor is through memory mapped I/O. The host could also be a custom-fit processor in its own right; thus, we also consider a host designed as a custom application-specific integrated circuit (ASIC) and analyze the impact of the efficiency of the host on the overall benefits of PCMOs. In Section 2.1.4.1 below, we introduce the metrics for evaluating PCMOs based architectures, which will also serve as a basis for comparison with conventional CMOS based architectures.

#### 2.1.4.1 Metrics for evaluating PCMOS based architectures

The two basic characteristics of interest in realizing efficient application-specific SoC architectures are the performance (typically the running-time of the application) and its energy consumption (or its derivative, power). To compare the efficiency of conventional SoC architectures, and PCMOS based SoC architectures, we introduce metrics based on these criteria. Our primary metric for consideration is the *energy-performance* product of an architecture which implements a particular application.

**Energy-Performance Product (EPP):** EPP is defined as the product of the application level energy (measured in Joules) and performance (measured in number of cycles).

Given the EPP of two alternate implementations—for example, the case when the entire algorithm is implemented as software executing on the host referred to as the baseline, compared to the case where the deterministic part of the algorithm is executed on the host with the probabilistic part executing on a PCMOS co-processor—they can be compared by computing the ratio of their individual EPP values. Since our goal is to compare the energy and performance gains realized through using PCMOS technology, we refine this notion and define the metric: EPP gain, which is denoted as  $\Gamma$ , and defined as follows.

**EPP Gain ( $\Gamma$ ):** EPP gain, denoted  $\Gamma$ , is the ratio of the EPP of the baseline to the EPP of a particular implementation. The EPP gain of a particular implementation  $I$  is determined as

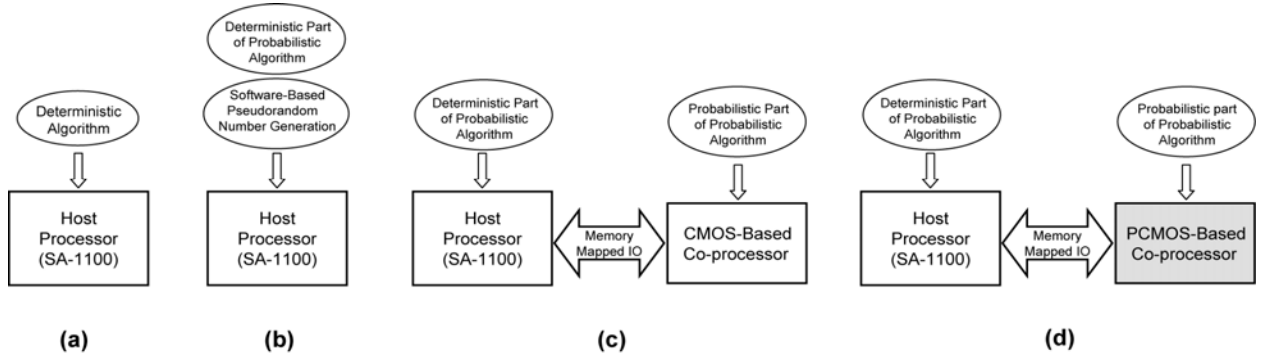
$$\Gamma_I = \frac{Energy_B \times Time_B}{Energy_I \times Time_I} \quad (2)$$

In Equation 2, the *baseline* denoted as  $B$  refers to the case when the entire application is realized using software on the host (for example, a StrongARM sa-1100 processor) only, without recourse to a co-processor. Thus, the numerator of  $\Gamma_I$  is derived for the case where the entire application executes deterministically on the host. The corresponding architecture realization is shown in Figure 9(a). While the baseline and hence the numerator of the EPP gain metric has been a purely deterministic realization of a deterministic algorithm corresponding to the case shown in Figure 9(a), in the context of applications that do not implement a deterministic algorithm, the software based emulation (illustrated in Figure 9(b)) shall serve as the baseline. We adopt this approach whenever the deterministic realizations do not exist or are impractically inefficient. In this case, the probabilistic component of the application is “emulated” using pseudo-random bit generation in software (as shown in Figure 9(b)).

A further refinement of this approach is to consider a co-processor (Figure 9 (c)) wherein the probabilistic parts of the application is emulated using a customized co-processor—typically using a pseudo-random number generator (PRNG, see Park and Miller [36]). Finally, as shown in Figure 9(d), the co-processor and hence the probabilistic computational component is realized using PCMOS. These cases (shown in Figures 9(a), 9(b), 9(c), and 9(d)) capture all reasonable alternate implementation scenarios against which the benefit of PCMOS technology is compared.

#### 2.1.4.2 Experimental methodology

The utility of PCMOS technology at the application level for computing platforms will be demonstrated by using the metrics (described in Section 2.1.4.1) and by considering a wide range of alternate implementations illustrated in Figure 9. Our experimental methodology, described herein is used to characterize these alternate implementations



**Figure 9:** The four possible realizations of an application using an SoC platform wherein (a) a deterministic application is executed entirely on the host, (b) a probabilistic application is executed entirely on the host, using an emulation based on pseudo-random bits generated using software, (c) the emulation is realized using a custom CMOS co-processor, and (d) a functionally identical PCMOS co-processor is used to realize the probabilistic components of the application

based on the metrics introduced. The primary metric of interest, the EPP metric, involves performance and energy estimation for each of the cases considered.

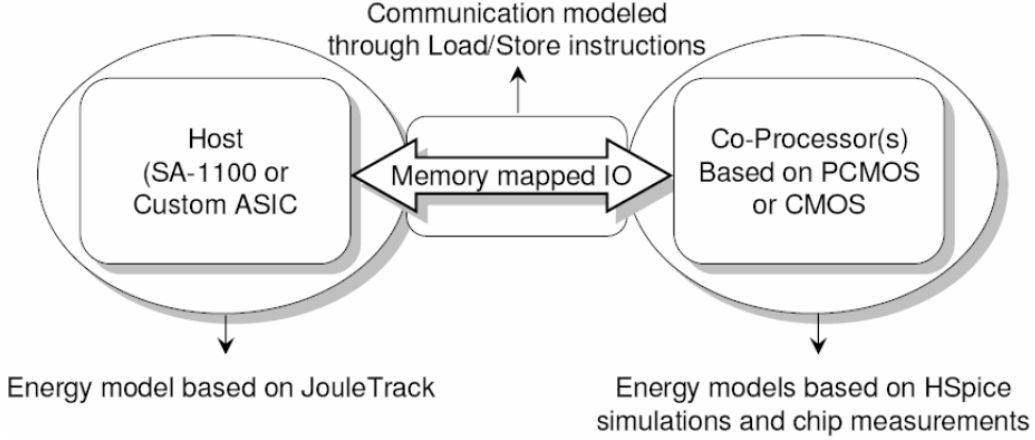
**Performance Estimation:** The performance of PCMOS and CMOS based SoC implementation and that of the baseline (where there is no co-processor) is estimated by using a modified version of the impact simulator of the Trimaran [8, 46] infrastructure. The modified simulator measures the performance of an application (the cycle count) executing on the StrongARM SA-1100 host. In addition, the simulator records a trace of the activity of the PCMOS and CMOS co-processors. This is combined with the performance models of co-processors, typically obtained through HSpice simulations, to yield performance in terms of execution time.

**Energy Estimation:** Three components of energy consumption are estimated. The energy consumed by the host, the energy consumed by the PCMOS (CMOS) based co-processor(s), and the energy cost of communication between the host and the co-processor(s). Since the co-processors are memory mapped, communication is through load-store instructions executed on the host. To quantify the energy consumed by the SA-1100 host, the JouleTrack model introduced by Sinha and Chandrakasan [41] is used. The performance and energy modeling techniques applied to various components of the SoC architecture are illustrated in Figure 10. The CMOS based co-processor involves a 32-bit pseudo random number generator (PRNG) [36] that is designed and synthesized into a TSMC 0.25 $\mu$ m process and its energy cost is derived from HSpice simulations. In the context of extensions based on PCMOS, the energy cost of the co-processor is derived from HSpice simulations as well as chip measurements of functioning probabilistic switches realized in TSMC 0.25 $\mu$ m process.

#### 2.1.4.3 Metrics for analysis of PCMOS based implementations

Consider a probabilistic application, and the three approaches to implementing it identified in Figures 9(b), 9(c), and 9(d)—wherein the baseline corresponds to the implementation shown in Figure 9(b).

We observe in a preliminary way that for these benefits to be achieved, significant



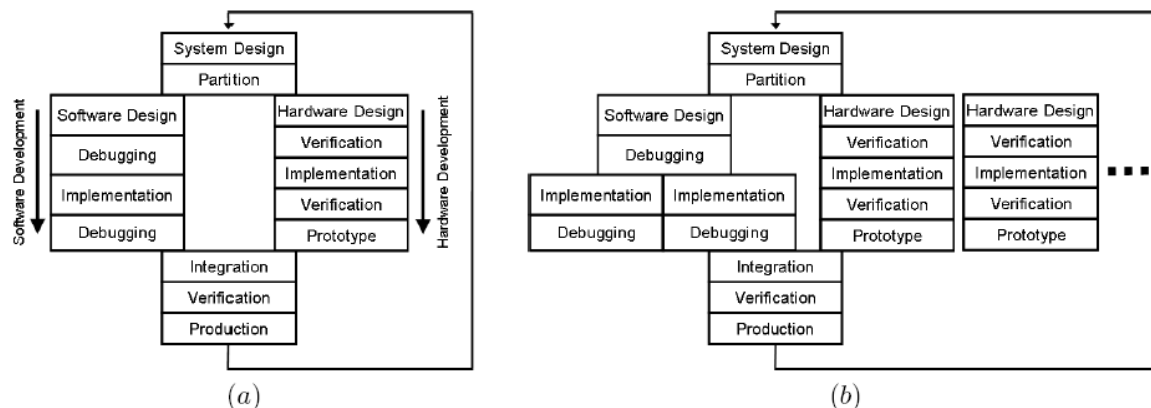
**Figure 10:** The host and co-processor architecture and power modeling

component of the application’s overall computational effort ought to be probabilistic (and hence, leverage the PCMOS based co-processor). We extend this central notion to define the flux of an application that quantifies the “opportunity” in an application to leverage PCMOS technology. Intuitively, the opportunity for the overall EPP gain depends on the proportion of the probabilistic operations in an application and the gain per probabilistic operation. We have characterized the former as flux. The latter is captured through the technology gain metric.

### 2.1.5 PCMOS based architectures for error tolerant applications

We also showed that besides probabilistic designs targeted for inherently probabilistic applications, PCMOS approach also offers significant benefits in the context of error-tolerant applications. This work was published in CASES’06 conference [17]. Examples for these applications are those from the signal and image processing domain, where hardware errors inherently reveal themselves as the signal-to-noise ratio (SNR) at the application level. Here, the application level quality, measured through SNR metric, can be traded against energy savings.

In the context of these error-tolerant applications, we built PCMOS-based arithmetic primitives such as adders and multipliers; and signal processing elements such as FIR filters and FFTs derived from them [17]. The resulting architectures are probabilistic and yet they compute the end result with adequate application quality. Moreover, we demonstrated that it is possible to trade amount of error or equivalently the SNR degradation against energy through the novel approach that we refer to as biased voltage scaling, or BIVOS. In this approach, significant effort is expended in computing the “more” significant bits and as a result, they are correct with higher probability—whereas the bits of lower significance are largely ignored. Conceptually, this “biased” probabilistic design methodology favors the most-significant components of a computing primitive such as a filter that contribute to a critical path with longer delays, and are more likely to affect the quality (accuracy) of its output through bit-significance. As an example, this can be accomplished by boosting the voltage ( $V_{dd}$ ) in going from the bits of lower significance to those of higher significance. Thus, bits of lower significance are permitted to be erroneous with a higher probability.



**Figure 11:** Typical design flow for (a) embedded computing systems, and (b) heterogeneous embedded computing systems

## 2.2 High Productivity Embedded Computing Technologies

We conducted a survey of heterogeneous embedded computing systems (HECS) in three main domains: namely, architectures, tools, and applications. Our goal was to survey existing technologies for HECS, and then categorize and analyze the individual domains, and finally, identify key bottlenecks and justify areas for tools and compiler technology for design and use of HECS. The results of the survey and analysis will be presented in Section 3.2. We will next present our methodology in analyzing the design flow for HECS.

In a typical embedded systems design flow, human expertise is both a critical ingredient and a costly productivity sink in the development process. Considering the typical design flow proposed by Handel Jones [19] (shown in Figure 11(a)), a system is first designed and partitioned into hardware and software requiring knowledge of different specification languages for each. Both hardware and software portions must then undergo a second iteration of design, followed by functional testing, implementation, a second round of testing (and finally prototyping in the case of hardware), with separate development teams for each. Once the hardware and software portions are complete, they must be integrated using vendor provided APIs and device drivers and tested once more on functional, transaction level, and timing simulators before the system can move to production. In all stages of the design flow, a high degree of human expertise is necessary. For HECS, design further complicates the development process by introducing multiple varied targets for design (Figure 11(b)).

To this end, we proposed automation through a parameterized architecture model to increasing productivity in HECS design. In this approach, an architecture is broken into tiles (such as a processor or memory element) and each tile has associated parameters. A compiler then uses the parametric description to compile a software solution for the entire HECS, as opposed to compiling for individual tiles with a designer performing partitioning and integration. By applying a compiler centric approach, human interaction in the design flow can be greatly reduced.

### 2.2.1 Parametric machine model

A central component of the automation process is the system modeling necessary to realize the high level compiler (HLC), hardware aware layer (HAL), and integrated simulation. This requires proper “hooks” for tuning, accuracy, and flexibility at each of these layers. The solution lies in parametric models that abstract key performance and power determinants allowing characterization of architecture and application properties. By exposing hardware details to a hardware aware HLC, human intervention can be reduced in the design cycle. The result is a parametric system model where attributes are defined for five individual tiles (programmable, configurable, memory, interconnect, and application specific tiles) of a typical HECS architecture.

The advantages of a parametric model for HECS can be summarized as follows:

- Design space can be reduced with correct parameter selections given (known) constraints/limitations
- Parametric representation is an abstract (high level) layer that can be mapped to compilers and the HECS design model
- Impose design choices (to be decided by designer/human) on parameters so as to help reduce design complexity
- Rule out design alternatives that do not comply with constraints
- Identify which parameters of one tile can or cannot be matched with which parameters of other tiles

E.g., 10Gb/s data rate, box-to-box communication over 5km distance dictates selection of high speed SAN interconnects, such as infiniband or RapidIO

- Parametric model can help integrated software-hardware development

## 2.3 *Reverse Engineering of Legacy Applications*

As mentioned in the Introduction, we conducted a survey to answer the question of “what is legacy code?” and then, we identified the hurdles of legacy code, and the current approaches used to deal with legacy code. In the sequel, we will briefly list answers to these research questions.

*What is legacy code?*

- *Definition 1:* A historical code that is known as working previously
  - Large, difficult, unfamiliar, complex code bases
  - Partially developed by multiple contributors
  - Code inherited from someone else
  - Incrementally developed within a long time frame
  - Different versions or upgrades exist
  - Code inherited from an older version
  - Code running on an obsolete hardware or platform



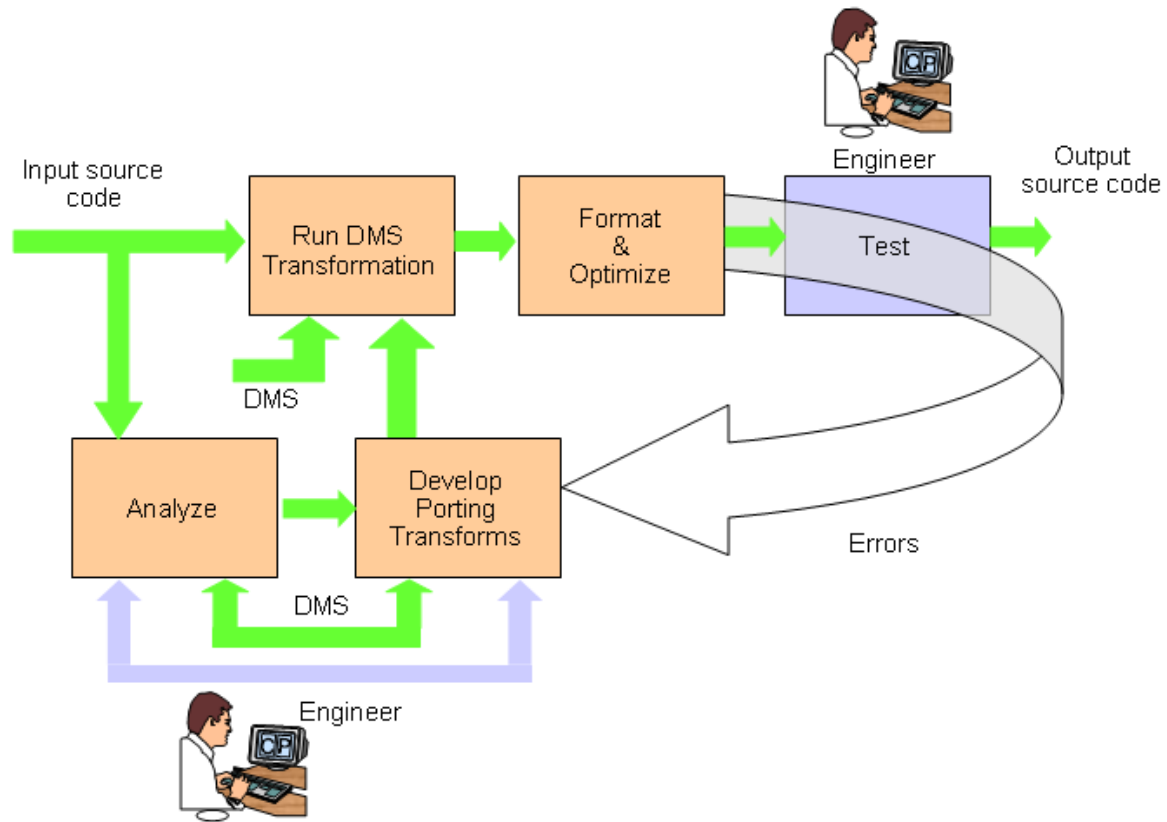
- *Definition 2: Code without adequate automated tests*
  - by Michael Feathers, Working Effectively With Legacy Code
  - “The main thing that distinguishes legacy code from non-legacy code is tests, or rather a lack of tests.”
  - Refers to the fact that legacy code is difficult to work with due to lack of tests (e.g., regression tests used to verify correct behavior, characterization tests used to detect and record modifications or to make sure that new modifications will not cause undesired consequences)
- *Definition 3: Code that you’re afraid of*
  - by Jeremy D. Miller, Lessons Learned for Dealing with Legacy Code
  - “Legacy code is code that you’re afraid of, but is too valuable or big to toss away”
  - Not necessarily an old code (most of it is less than 18 months old)
  - We do not understand the code, i.e., we have very limited specifications on functional behavior of the code. Therefore, it is hard to debug, expand, improve and deploy
  - The feedback cycles are too slow (i.e., the elapsed time between writing a line of code and knowing that the line of code works)

### *Hurdles of legacy code*

- Reliability
  - Testing and fixing bugs are hard
- Maintainability
  - Adding features
  - New application development
  - Reuse
- Migration
  - Portability across platforms and languages

### *Current approaches for dealing with legacy code*

- Language and file format conversion tools
  - Examples from Siber Systems:
    - DataReaders (converts Cobol data files to modern formats such as CSV, DBF, Excel, Oracle, etc.)
    - CobolTransformer (analyzes, converts, or generates Cobol source programs)
- Tools for understanding
  - Code structure
  - Dependencies
- Automated steps for analysis, formatting, optimizing, and porting transforms
- Manual interaction also needed for generating an acceptable output code (e.g., for using automated steps efficiently and for testing resulting output code)



**Figure 12:** Example for state-of-the-art tools by Semantic Designs, Inc.

We show one example state-of-the-art technology for dealing with legacy code in Figure 12. The figure illustrates a Design Maintenance System (DMS) from Semantic Designs, Inc. A set of tools are used for automated analysis, modification, and transformation. Their approach supports a variety of input and output languages, as well as mixed language input, so as to enable working with more than one input language domain simultaneously. Their approach also supports databases. Other characteristic of their approach is that it uses configurable transformation and analysis rules as well as output code formatter and optimizer. It can also generate compilable output code and it is scalable to thousands of input files and millions of lines of code.

In short, the main hurdles for dealing with legacy code are due to the semantic gap between the user and legacy code (lack of documentation and programmer intention), lack of test benches, and the language that legacy code is implemented in. In the case of language of legacy code, problems arise due to the fact that it restricts re-use and enhancement of the existing framework, and also problems occurs with porting. Finally, one significant hurdle is, again, the absence of tools for debugging, optimization, and visualization.

We will introduce our proposed approach in Chapter 3.3.

## **2.4    *Software Standards and Techniques for Cognitive and Commodity Computing Systems***

### **2.4.1    ACIP living framework forum task**

The original goal of the ACIP LFF task was to develop an actionable plan for a joint effort of the ACIP prime contractors to develop a cross-platform software development approach that provides portability, high performance, and ease of use. The development approach was intended to interface to the Morphware Stable Interface (MSI) environment that was being developed under the DARPA PCA program.

The original technical approach for this task was to develop a plan for the Living Framework Forum by attending all ACIP Principal Investigator (PI) meetings; seeking to develop an understanding of individual ACIP project software development environments; identifying common approaches between projects as well as areas of divergence; and proposing an approach for a common software environment and a plan for developing the required standards, specifications, benchmarks, and software. In keeping with this tasking, Georgia Institute of Technology (GT) personnel attended the 2nd (May 11, 2005) and 3rd (November 2, 2005) ACIP PI meetings. At these meetings, GT personnel discussed software architecture issues with representatives of each of the three prime ACIP development teams, as well as with other groups funded by ACIP for technology development projects. In addition, GT participated in the AFRL/Cornell Cognitive Architectures Workshop held in July 2005. GT also planned a series of individual site visits to each of the three prime ACIP teams. The first visit was held with the Lockheed Martin ACIP project team February 28, 2006, to discuss their software approaches for ACIP.

The GT team met with DARPA on March 7, 2006 to discuss initial findings and plan activities for the remainder of ACIP phase 1. At this time, GT was directed by DARPA to change the task focus to participation in the ACIP Working Group (ACIP-WG) RECENTLY convened by DARPA. The purpose of the ACIP-WG was to develop a consensus on what constitutes the target ACIP cognitive domain (compact applications), the programming approach for ACIP systems (cognitive API), and the evaluation methods for ACIP systems (benchmarks and metrics). GT focused particularly on the “compact applications” and “kernel merging/taxonomy” sub-groups.

GT attended and participated in the April 20 and May 25-26, 2006 ACIP-WG meetings, providing input to the ACIP-WG plan at those meetings. At that point, funding for this task was exhausted and no further activity was conducted.

### **2.4.2    GPU SVM task**

The initial goal of the GPU SVM task was to develop, test, and evaluate an implementation of the PCA SVM API for commodity GPUs. Such devices provide a good example of a high performance processor that is a candidate for inclusion in heterogeneous systems. The task was expected to demonstrate the suitability of portable virtual machine (VM) layers for targeting GPU platforms, and aid in the development of the SVM by elucidating needed extensions to include GPUs as target devices.

The initial technical approach for this task was to develop an SVM library for standard desktop workstations with commodity GPU boards. The library would allow the Reservoir

Labs R-Stream high level compiler (HLC) to manage memory in the main system and the GPU, and allow SVM kernel work functions to be overridden with openGL fragments written in the Cg language. Complemented by a Low Level Compiler (LLC) developed by Reservoir Labs that translates kernel work functions to Cg, the library would allow SVM programs running on targeted systems to make use of the GPU to accelerate execution. Once the library and LLC were developed, portions of the One Semi Automated Forces Objective System that are particularly suited to stream processing would be ported to the HLC input language, known as “Gumdrop”, by compiling them through the Morphware tool chain and benchmarking them with respect to various performance metrics. Throughout the process, in coordination with Reservoir, GT was to evaluate the Morphware Stable Interface (MSI) elements and make suggestions to the Morphware Forum regarding improvements to the MSI. The task focused specifically on clarification and augmentation of the SVM specification, improvements in HLC and LLC interaction and feedback, and approaches for platform configuration control within the MSI.

## CHAPTER 3

### RESULTS

#### 3.1 *Power Aware Computing Through Management of Computational Entropy*

One significant result of our work is that *while the energy consumed by deterministic switching is never less than  $(-\kappa t \ln 2)$  Joules, referred to often as the “fundamental (or thermodynamic) limit”; in contrast, the energy consumed by an idealized probabilistic switch with an associated probability of error of  $(1-p)$  is lower, and can be as low as  $(-\kappa t \ln 2p)$  Joules for each switching step.* Here,  $\kappa$  is the well-known Boltzmann’s constant,  $t$  is the temperature of the thermodynamic system, and  $\ln$  is the natural logarithm. This result established that at the fundamental limit, probabilistic algorithms offer the potential for energy savings of  $\kappa t \ln(1/p)$  Joules per primitive switching step. In keeping with traditional idealizations, our switches are not lossy since switching is always performed at thermal equilibrium.

Principles of statistical thermodynamics may be applied to switches (as described in Chapter 2.1.1) to quantify their energy consumption, and hence the energy consumption (or energy complexity) of a network of such switches. To reiterate, while a switch that realizes the deterministic non-trivial switching function consumes at least  $(-\kappa t \ln 2)$  Joules of energy [31], a probabilistic switch can realize a probabilistic non-trivial switching function with  $(-\kappa t \ln 2p)$  Joules of energy where  $p$  is the probability parameter [33]. The complete definition of a probabilistic switch, the operation of a network of probabilistic switches and a derivation of energy complexity of such networks were published in IEEE Transactions on Computers in 2005 (please see [33]).

##### 3.1.1 Analytical model and the three Laws of a PCMOS inverter

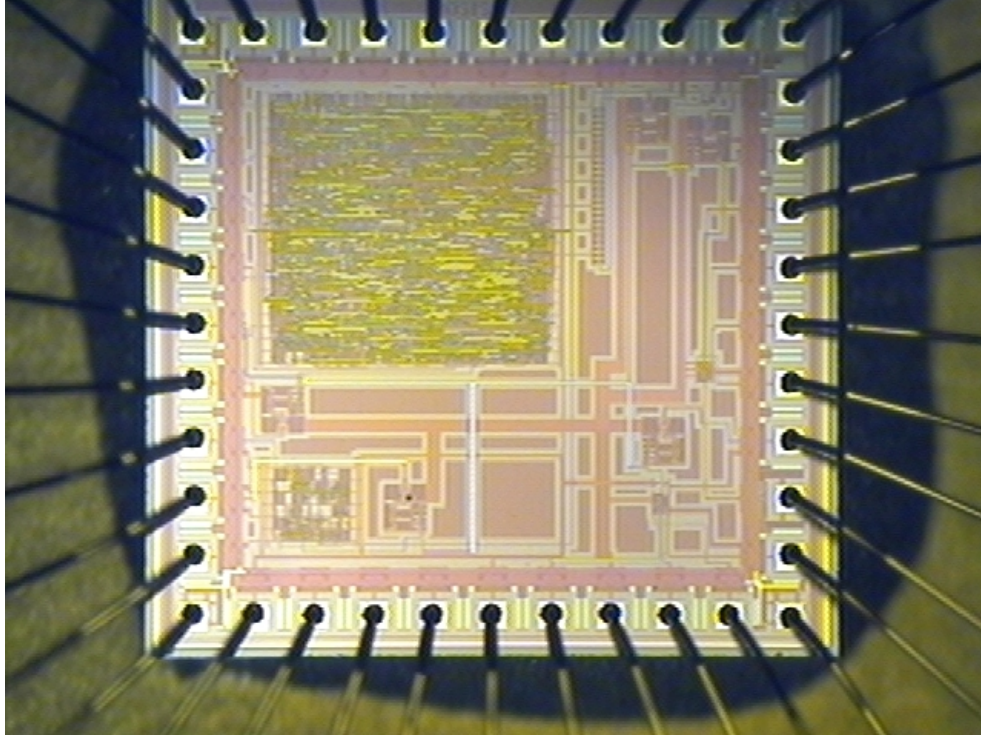
The analytical model of a PCMOS inverter is given in Equations (3), (4) and (5) below [43, 1, 11, 23]. This model is derived from the output voltage distribution (see Figure 6(c)) of an inverter coupled with thermal noise at its output (see Figure 6(a)) and summarizes the relationship between the probability  $p$ , the operating voltage  $V_{dd}$ , and the noise magnitude  $\sigma$ .

##### Analytical Model of a Probabilistic Inverter

$$p = 0.5 + 0.5 \cdot \operatorname{erf}\left(\frac{V_{dd}}{2\sqrt{2} \cdot \sigma}\right) \quad (3)$$

$$\sigma = \frac{V_{dd}}{2\sqrt{2} \cdot \operatorname{inverf}(2 \cdot p - 1)} \quad (4)$$

$$E = 4 \cdot C \cdot \sigma^2 \cdot [\operatorname{inverf}(2 \cdot p - 1)]^2 \quad (5)$$

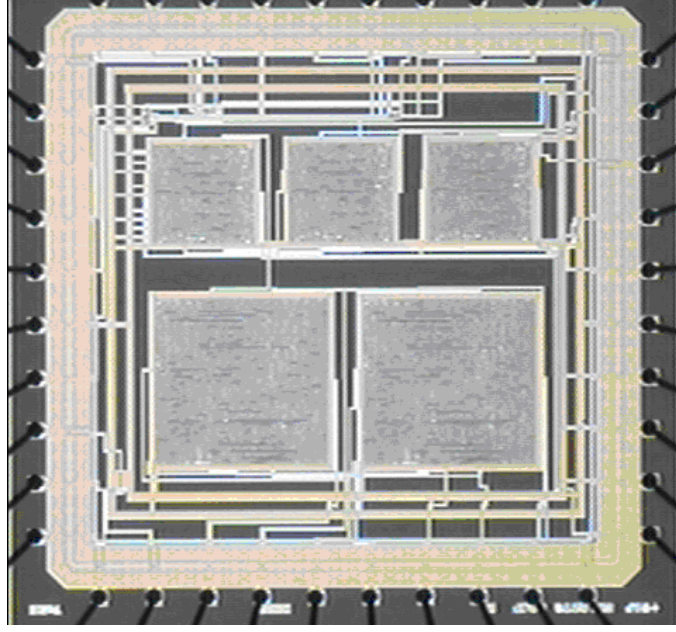


**Figure 13:** PC MOS chip fabricated in AMI 0.5 $\mu$ m technology

Note that in the above equations, *inverf* refers to the inverse of the well-known error function *erf* (see [44]) and *C* corresponds to the output capacitance of the inverter. As we demonstrated in our publications [1, 24], the error function behavior can be approximated using asymptotic notions from algorithm analysis in computer science. Briefly, *E*, the energy to produce a probabilistic bit (as seen in Equation 5), grows with *p* and the order of this growth dominates an exponential. Such an approximation of the above analytical model allows us to deduce relationships between the probability parameter *p*, the voltage, or equivalently signal magnitude  $V_{dd}$ , the noise magnitude  $\sigma$ , and finally, the energy consumed per switching step denoted by *E*. The relationships can be extrapolated over successive technology generations by substituting the corresponding load capacitance value *C* from the International Technology Roadmap for Semiconductors (ITRS) [20] in the model.

Based on the above analytical model, we characterized the behavior of a PC MOS inverter through the two laws that relate the energy consumed, the associated probability parameter *p* and the noise RMS  $\sigma$ . Moreover, we validated the laws and the corresponding analytical model via HSpice simulations using 0.5 $\mu$ m AMI, 0.25 $\mu$ m TSMC, and 90nm and 65nm IBM processes as well as via the physical measurements of PC MOS inverters implemented using 0.5 $\mu$ m AMI and 0.25 $\mu$ m TSMC processes that were available to us for fabrication. The chip photos for the 0.5 $\mu$ m AMI and 0.25 $\mu$ m TSMC processes are illustrated in Figures 13 and 14.

Briefly, the *first law* relates the energy consumed per switching step to *p*, given a fixed amount of noise magnitude  $\sigma$ ; whereas the *second law* relates the energy consumed per switching step to  $\sigma$ , given a fixed value of *p*. For a more complete description of the laws with asymptotic notions, please see our publications [1, 24].



**Figure 14:** PCMOS chip fabricated in TSMC 0.25μm technology

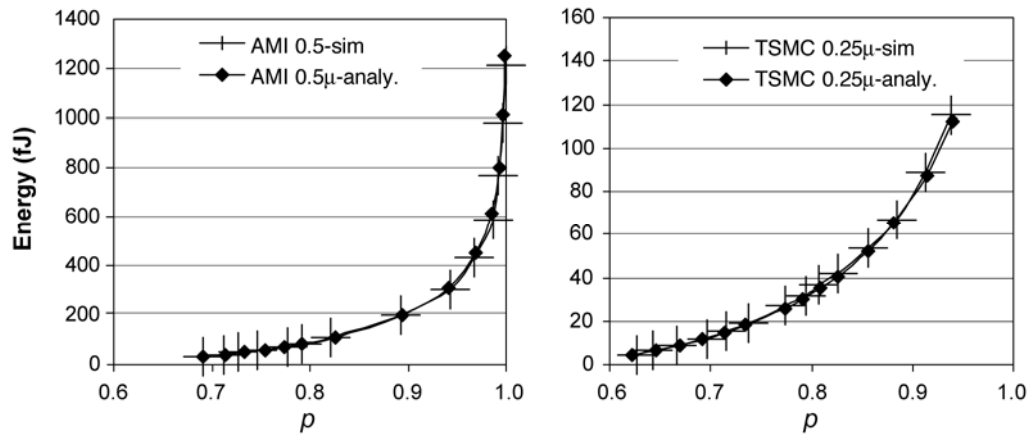
**Law 1** For any fixed technology generation or feature size (which determines the capacitance  $C$ ) and constant noise magnitude, the switching energy  $E$  consumed by a probabilistic switch grows with  $p$ . Furthermore, the order of growth of  $E$  in  $p$  is asymptotically bounded below by an exponential in  $p$ .

As shown in Figure 15, Law 1 captures the changes in  $E$  by varying probability  $p$  (resulting from varying  $V_{dd}$ ) value determined by Equation 5 of the analytical model. As shown in this figure, the analytically estimated values track simulated results well. Next, we present the second law.

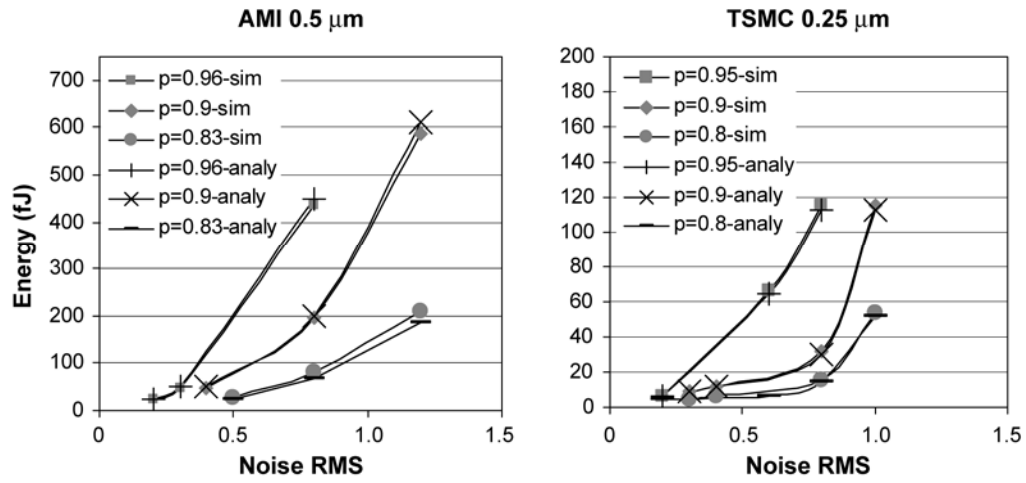
**Law 2** For any fixed technology generation (feature size), the switching energy  $E$  consumed by a probabilistic switch increases quadratically with noise magnitude, whenever  $p$  remains constant.

Law 2, illustrated in Figure 16, relates the quadratical changes in  $E$  to the variations in  $\sigma$ , value determined by Equation 5 of the analytical model. This quadratical relationship between  $\sigma$ , and  $E$  for a fixed value of  $p$  follows from the fact that the switching energy  $E = 1/2 \cdot C \cdot V_{dd}^2$  is quadratically related to  $V_{dd}$  which is linearly dependent on  $\sigma$  (as shown in Equation 4). Moreover, it is intuitive that as  $\sigma$  is increased, more energy will be consumed to realize the same value of  $p$ . Similar to the case of Law 1, and as seen from Figure 16, the analytical results are matched very closely with the simulation results.

The two laws presented above relate  $p$  and  $\sigma$  to energy and the relationships were obtained by varying the three independent parameters, namely, the operating voltage  $V_{dd}$  (referred to as the signal), the noise magnitude  $\sigma$  and the technology generations (e.g., 0.25μm and 0.5μm considered in this paper). These relationships were established through varying the value of  $V_{dd}$  and the value of  $\sigma$ . A more succinct form of the analytical model that characterizes  $p$  is shown in Equation 6 below. Here, rather than specifying the values of  $V_{dd}$  and  $\sigma$  as two independent variables, they are presented as a single ratio, which we

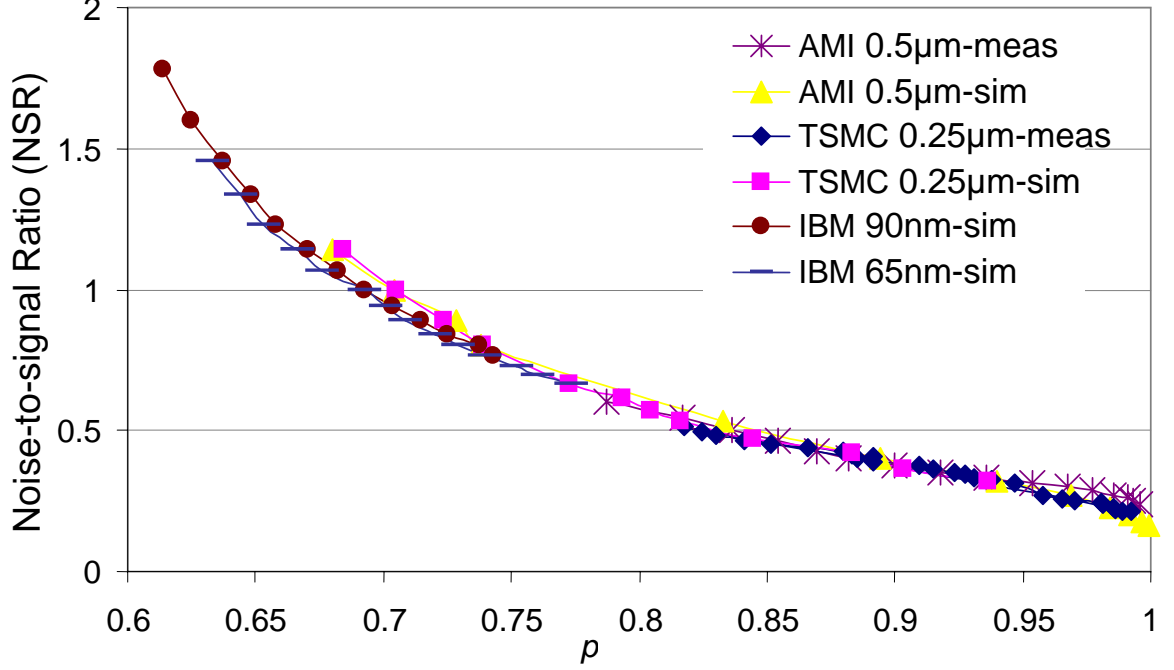


**Figure 15:** Law 1 validated over 0.5μm and 0.25μm processes.



**Figure 16:** Law 2 validated over 0.5μm and 0.25μm processes.





**Figure 17:** Validation of invariance law of PCMOS through simulations and measurements. The invariant behavior of  $p$ -NSR relationship is shown for four technology generations, 90nm, 65nm, 0.25 $\mu$ m and 0.5 $\mu$ m, based on HSpice simulations as well as physical measurements of PCMOS inverters fabricated using 0.25 $\mu$ m and 0.5 $\mu$ m processes.

refer to as the Noise-to-Signal Ratio (NSR). Given that  $V_{dd} = 2\sqrt{2} \cdot \sigma \cdot \text{inver}f(2 \cdot p - 1)$ , from Equation (4), we can express NSR as follows.

Noise-to-signal Ratio (NSR)

$$NSR = \frac{\sigma}{V_{dd}} = [2\sqrt{2} \cdot \text{inver}f(2 \cdot p - 1)]^{-1} \quad (6)$$

Because of the fact that NSR captures the two independent dimensions ( $V_{dd}$  and  $\sigma$ ) into a single dimension, we will refer the third law as the “unifying” law of a probabilistic inverter stated as in below.

Law 3 Independent of the technology generations (feature size), NSR uniquely determines the probability parameter  $p$ .

This form of the model of a probabilistic inverter is very interesting for the following reasons. First, it allows us to establish a succinct relationship between the independent parameters and the probability parameter  $p$  in a manner that does not depend on the CMOS technology generations; for example, there is no dependence on the capacitance  $C$ , which is typically determined by the feature size. Secondly, it is possible to simultaneously illustrate the invariance across technology generations, and easily estimate  $p$  given an NSR

value or vice versa. Figure 17 shows the NSR and  $p$  relationship. As seen from the figure, the data points from all the simulated processes (90nm, 65nm, 0.25 $\mu$ m, and 0.5 $\mu$ m) and from the measured processes (0.25 $\mu$ m and 0.5 $\mu$ m) perfectly overlap with each other and match the results of the analytical model. This validates the invariance law of PCMOS and confirms the NSR as a figure-of-merit that uniquely determines  $p$  independent from the technology generations.

### 3.1.2 Analysis and optimization of energy, performance, and probability of PCMOS circuits

We showed the design trade-offs between energy, performance, and  $p$  of PCMOS gates using analytical models of energy, propagation delay, and  $p$ . We have considered  $V_{dd}$  and  $V_{th}$  as our design variables and found the values of  $V_{dd}$  and  $V_{th}$  for optimal energy-delay product (EDP) and  $p$  under given constraints on  $p$ , performance, and EDP. We have observed that operating a PCMOS gate at lower supply voltages is more preferable to minimize its EDP. By contrast, for maximizing the  $p$  of a PCMOS gate, operating the gate at higher supply voltages is preferable. We observed that the optimal values of  $V_{dd}$  and  $V_{th}$  are contingent upon the constraints imposed by the application as well as the models used for energy, delay, and  $p$ . We also performed circuit simulations to validate our analytical models. From the simulations we have observed that the shapes of EDP surfaces and location of the optimal EDP point are dependent on the models used for energy, delay, and  $p$ . Our analysis can be helpful in circuit design for applications with specific  $p$ , performance, and EDP requirements.

We also included an analysis of the impact of parameter variations in threshold voltage, temperature, and supply voltage on EDP, performance, and  $p$  of PCMOS gates as well as on optimal values of EDP and  $p$ . We found that accurately estimating the variations in temperature, threshold voltage, and supply voltage is important for accurately optimizing the EDP and  $p$  of PCMOS gates. Furthermore, we briefly discussed the effect of the noise on the energy consumption of a PCMOS gate, as well as the effect of the noise duration on the  $p$  of a PCMOS gate. Noise causing spurious switchings may increase the energy consumed by a PCMOS gate. The ratio of the noise duration to the propagation delay of the gate is important in determining the  $p$  of a PCMOS gate. If the noise duration is smaller than the propagation delay of the gate, then high frequency components of the noise are filtered by the gate. Hence, we conclude that probability and energy models of probabilistic gates should consider these issues to accomplish accurate optimizations.

### 3.1.3 Results and analysis for probabilistic system on-a-chip (PSoC) architectures

To evaluate the efficiency of PSoC architectures, we investigated applications from the cognitive and embedded domain which embody probabilistic behaviors. Probabilistic algorithms are those in which each step, upon repeated execution with the same inputs, could have several possible outcomes, where each outcome is associated with a probability parameter. In particular, the applications we considered are based on Bayesian inference [39, 37, 3], Probabilistic Cellular Automata [14], Random Neural Networks [16, 15] and Hyper Encryption [13]. For brevity, these algorithms will be referred to as BN, PbCA, RNN and HE respectively. Common to these applications (and to almost all probabilistic algorithms) is the notion of a core probabilistic step with its associated

probability parameter. An abstract model of a core probabilistic step is a probabilistic truth table. Intuitively, realizing probabilistic truth tables using probabilistic switches is inherently more efficient with PCMOs switches than with conventional (deterministic) CMOS switches. This is because of the inherent probabilistic behavior of the PCMOs switches. When compared to a baseline implementation using software executing on a StrongARM SA-1100, the gain of a PCMOs based PSoC is summarized in Table 1. In addition, when the baseline is a custom ASIC realization (host) coupled to a functionally identical CMOS based coprocessor, in the context of the HE and PbCA applications, the gains are increased to 9.38 and 561, respectively. Thus, for applications which can harness probabilistic behavior, PSoC architectures based on PCMOs technology yield several orders of magnitude improvements over conventional (deterministic) CMOS based implementations. For a detailed explanation of the architectures, a description of the applications, and results, please see our publications [7, 9]:

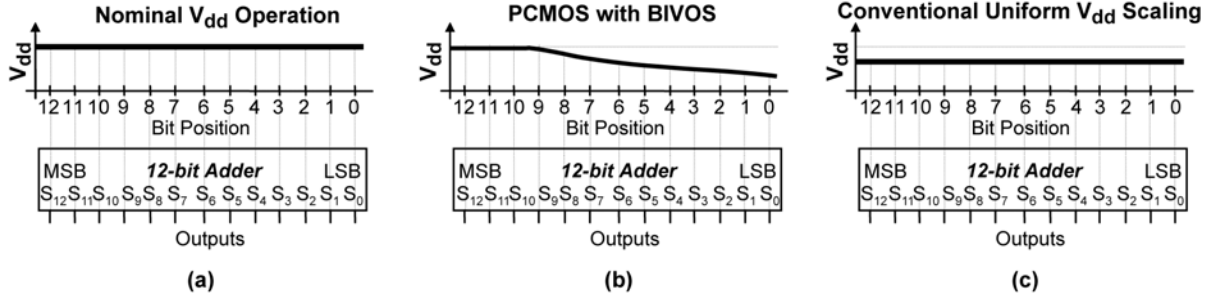
- Lakshmi N. Chakrapani, Bilge E. S. Akgul, Suresh Cheemalavagu, Pinar Korkmaz, Krishna V. Palem, and Balasubramanian Seshasayee, “Ultra Efficient Embedded SOC Architectures based on Probabilistic CMOS Technology,” *Proceedings of the 9th Design Automation and Test in Europe (DATE)*, March 2006, pp. 1110–1115.
- Lakshmi N. Chakrapani, Pinar Korkmaz, Bilge E. S. Akgul, and Krishna V. Palem, “Probabilistic system-on-a-chip architectures”, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, August 2007, Vol. 12, Issue 3.

Algorithm	$\Gamma_I$	
	Min	Max
BN	3	7.43
RNN	226.5	300
PbCA	61	82
HE	1.12	1.12



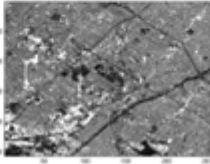
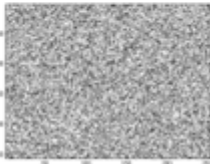
**Table 1:** Application level maximum and minimum EPP gains of PCMOs over the baseline implementation where the implementation  $I$  has a StrongARM SA-1100 host and a PCMOs based co-processor

### 3.1.4 BIVOS and DSP results

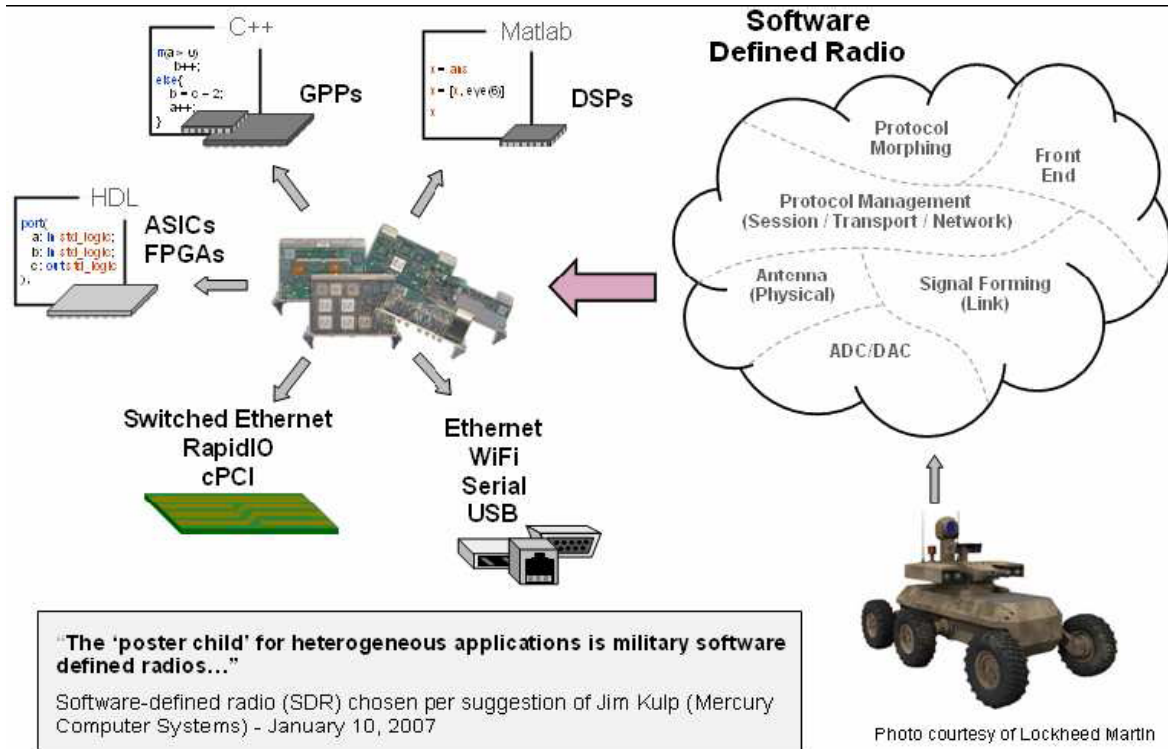
To compare the quality of the solution using our novel BIVOS design methodology with the conventional voltage scaling approach, we consider Figure 18—here, the voltage is lowered uniformly across all of the bit positions of the arithmetic primitives. The associated energy savings and the associated image quality for both the H.264 application (using a PCMOs based fir element) and the synthetic aperture radar (SAR) imaging (using a PCMOs based FFT element) are illustrated in Table 2. It is easily seen from Table 2 that the quality of the solution is significantly worse in the case of conventional voltage scaling when compare to our novel approach (Figure 18(b)), wherein there is (almost) not discernible visual degradation and is comparable to the original image computed using entirely conventional digital hardware with much higher energy consumption. More details of this work is published and presented at CASES’06 conference [17]. This work on probabilistic arithmetic and BIVOS approach appears in our patent application filed on February 2006 [34].



**Figure 18:** (a) Conventional digital design with very high (nominal) voltage levels. (b) Our probabilistic, BIVOS approach with significantly lower energy consumed and leasing to minimal degradation of the quality of the image. (c) Conventional voltage scaling that achieves the same level of energy savings as (b) but with significantly lower image quality.

	H.264 Image Compression		SAR Imaging	
	PCMOS with BIVOS	Conventional voltage scaling	PCMOS with BIVOS	Conventional voltage scaling
Energy savings	2X	2X	1.6X	1.6X
Image				

**Table 2:** Comparing the quality of the output images achieved through the novel BIVOS based PCMOS to that achieved through conventional voltage scaling and the corresponding energy savings of both approaches when compared to nominal operation at full-scale  $V_{dd}$ .



**Figure 19:** A canonical heterogeneous computing system

## 3.2 High Productivity Embedded Computing Technologies

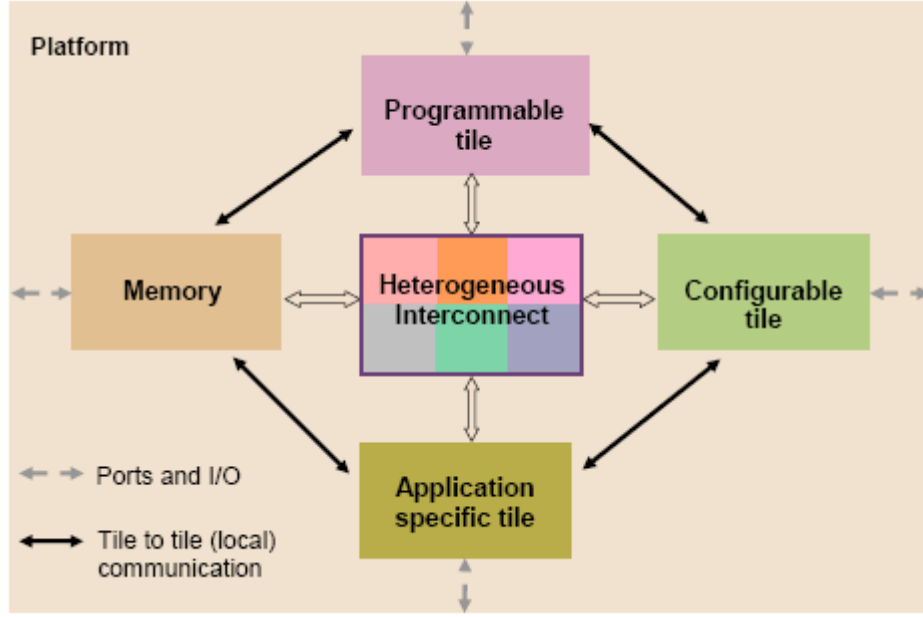
We have identified the shortcomings in development and productivity of heterogeneous embedded systems design; and proposed solutions and research directions to remedy these shortcomings. Here, we will present the results of our survey, introduce the productivity challenges and need for tools in design and use of HECS, and show our proposed compiler centric approach to massively improved productivity.

### 3.2.1 Survey of HECS

A canonical heterogeneous computing system is shown in Figure 19. This example constitutes a military software-defined radio with a variety of computing elements and subsystems including antenna, protocol management, signal processing, etc. Such an HECS involves three main domains: architectures that comprise the physical hardware components, tools that are used for design and development of an HECS, and applications. Each domain has its own attributes, metrics, and classifications. We will next present the results of our survey for each domain and their attributes.

#### 3.2.1.1 Architectures

An HECS architecture platform, that is composed of programmable tile(s), memory tile(s), application specific tile(s), configurable tile(s), and interconnects, is shown in Figure 20. A platform has a set of attributes used to analyze and classify a platform and to interrelate



**Figure 20:** Model for heterogeneous embedded system architectures

an architecture platform to other HECS domains (such as tools and applications). The definition of attributes for individual tiles plays a critical role in parameterizing a platform to cope with the rich design space of a typical HECS.

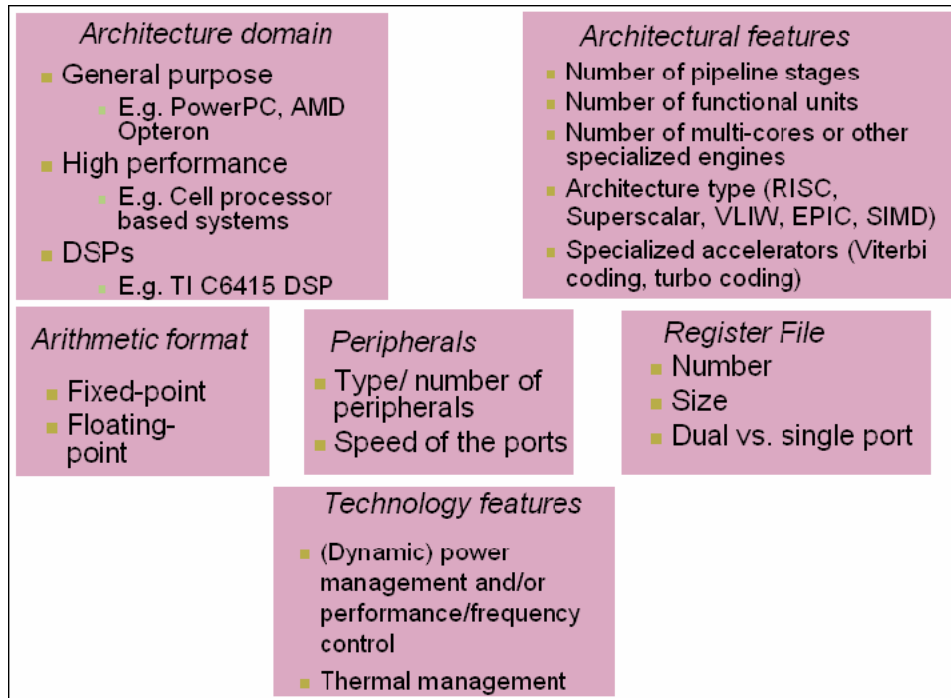
These (architecture) platform attributes are performance (measured in GFlops), power consumption (measured in Watts), performance per power – reflecting efficiency – (measured in terms of GFlops per Watt), configurability level (measured by the number of configurable elements/tiles), parallelism level (measured in the number of computing resources and corresponding silicon area), and scalability level (measured in terms of interconnect types and the number of modules that can fit into a box or board). Other attributes measured qualitatively can be listed as follows: Ease of redesign, reusability depends on scalability and configurability of all the tiles; application domain specific attribute as defined by applications and drive use of application specific accelerators; real-time attribute depends on performance and application-domain specification; and security depends on level of re-configurability and middleware/OS support. Some attributes are interrelated to others: e.g., performance depends on parallelism and power consumption, whereas redesign and reusability depend on configurability and scalability.

The attributes also affect the choice and use of tools. Moreover, each platform tile also has its own tile-specific attributes that are used to parameterize an HECS architecture (see Section 2.2.1 for a discussion on our parametric machine model of an HECS). The corresponding attributes of one example tile (programmable tile) of a platform and how they relate to platform attributes are respectively shown in Figures 21 and 22. Similarly, we identified attributes to parameterize interconnects and memories as well.

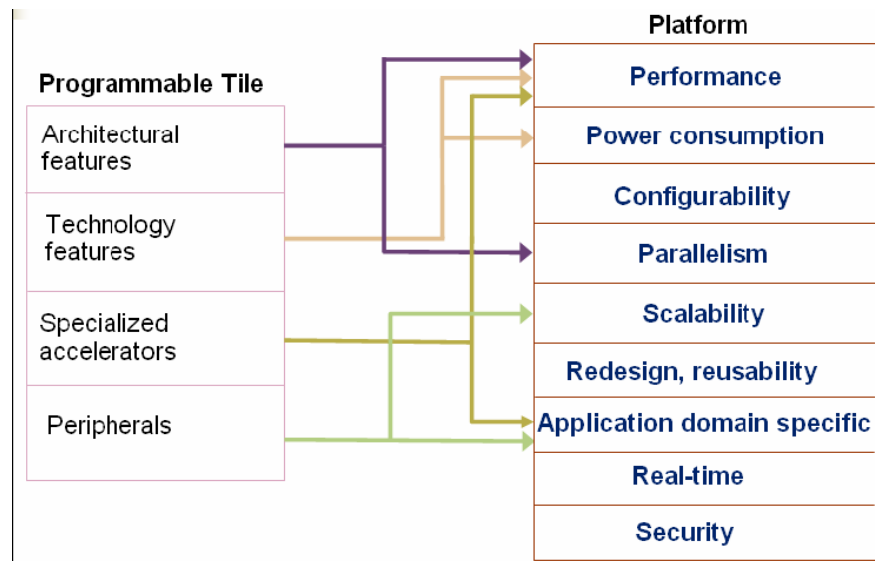
These attributes help us to compare and relate platforms. Among various platforms that we studied in our survey, we chose five examples to illustrate in Figure 23.

The specifications of our example platforms are listed as follows:

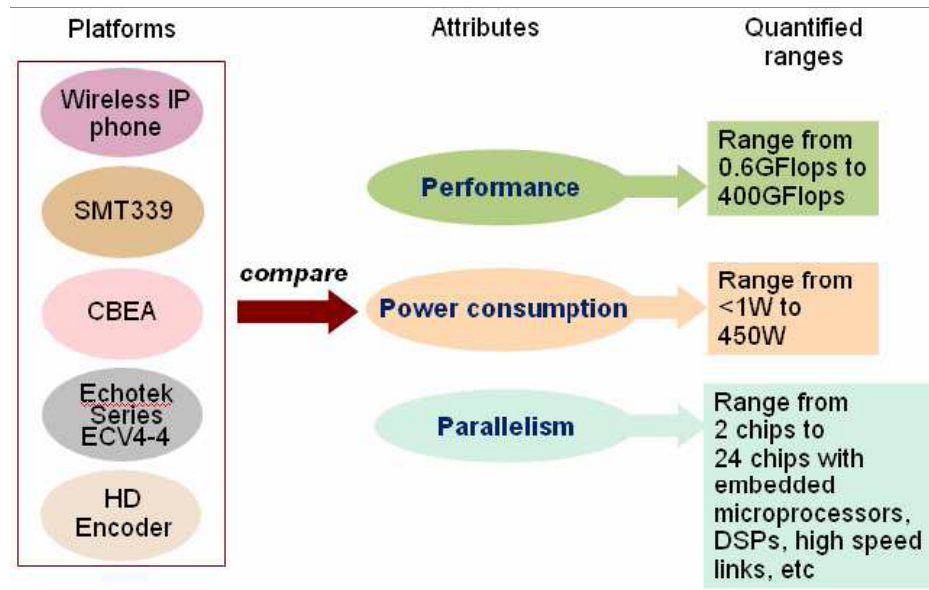
- Wireless IP phone evaluation platform from Renesas



**Figure 21:** Attributes of a programmable tile



**Figure 22:** Relation between programmable tile attributes and platform attributes



**Figure 23:** Comparing platforms with their quantifiable attributes

One Reduced Instruction Set Computer (RISC) processor from Renesas, one Field Programmable Gate Array (FPGA) from Quicklogic

Low power, configurable, real-time

- SMT339 High speed image processing module from Sundance

One TIDM642 DSP, Xilinx FPGA including embedded PowerPC cores

High performance, reconfigurable, scalable, real-time

- CBEA (Cell Broadband Engine Architecture)

Eight Synergistic Processor Elements (SPEs), One Power Processor Element (PPE), high speed Element Interconnect Bus (EIB) interconnect  
High performance, parallel, scalable, real-time

- Echotek Series ECV4-4 digital receivers from Mercury

Eight Xilinx and Altera FPGAs

Reconfigurable, application domain specific, real-time

- High definition (HD) encoder from Scientific Atlanta

Four TIC6415 DSPs, multiple (up to 20) Xilinx FPGAs, multi-gigabit links

High performance, parallel, scalable, reconfigurable, real-time

As observed from these examples, a variety of processors, FPGAs and ASICs exist on the same platform. The lessons learned from our survey can be summarized as follows: Given the complexity and heterogeneity of a typical HECS, application partitioning becomes extremely critical both in terms of design and use of HECS. Performance analysis will help to perform efficient partitioning to overcome bridging/bottleneck problems. We also observe that most of the DSP functionality migrates to FPGAs, which implies need



for tools: Existing DSP codes written in C need to move to hardware description language (HDL), hence new design flow emerges, and so does need for DSP-FPGA codesign tools. Moreover, advanced parallel architecture design (e.g., in CBEA) increases programmers role in optimizing code to efficiently exploit architecture features and program so as to parallelize applications. We also see dedicated processors (e.g., PPE in CBEA) running operating system (OS) and other control intensive programs, specialized architecture features, and the need for highly optimized software (math, DSP, I/O) libraries. On the other hand, these HECS platforms include multi-target, heterogeneous “malleable” and “non-malleable” elements. This necessitates compilers and software development tools to partition, parallelize, and map applications to hardware, target specific optimizations for performance and power, improve and automate code portability. Also, modeling tools are needed to enable integrated hardware-software development, verification and debugging, and faster time-to-market. Finally, a standardization for cross-compatibility between tools is needed.

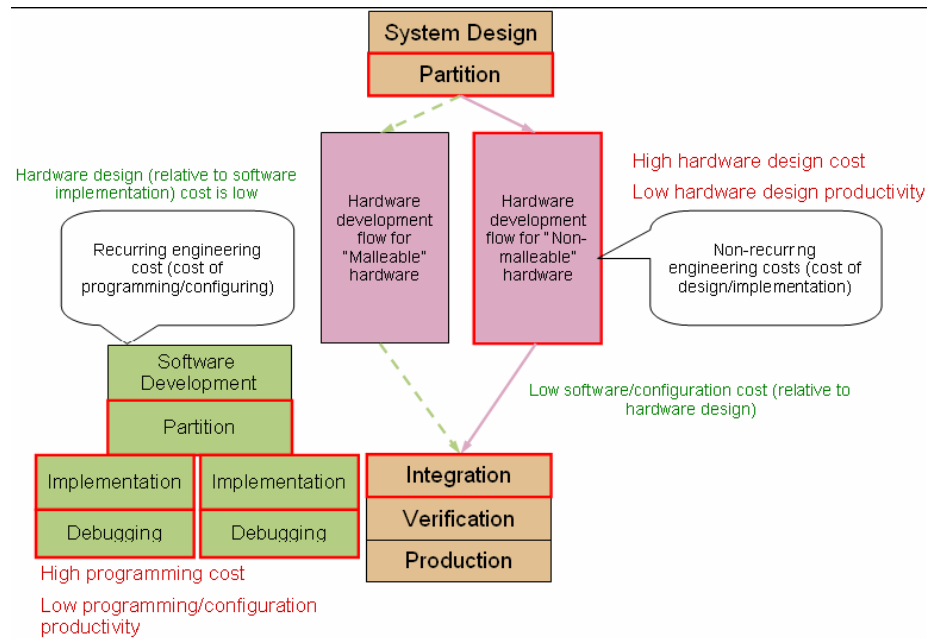
### 3.2.1.2 Tools

We can understand the importance of tools in HECS design and use by recalling the HECS design flow shown in Figure 24. After system design, partitioning design into hardware components involves two main paths: hardware development flow for malleable hardware and hardware development flow for non-malleable hardware. In existing technologies for HECS design and use, non-malleable hardware development has low hardware design productivity and high hardware design cost, wherein non-recurring engineering costs (cost of design and implementation) emerge. However, and as expected, the development for non-malleable hardware will constitute a lower software/configuration cost (relative to hardware design) and the hardware design (relative to software implementation) cost is low. As for the malleable hardware development, however, hardware design (relative to software implementation) cost is low. In the software development context, on the other hand, we observe low programming/configuration productivity and high programming cost, wherein recurring engineering cost (cost of programming/configuring) emerges.

We categorized heterogeneous systems into three types of systems that are being designed: Heterogeneous platform (type 1) with multiple, heterogeneous boards and interconnects; heterogeneous system on a chip (HSoC) plus FPGA (type 2); and HSoC (type 3). Among these types, our survey results show that tools for heterogeneous platforms (type 1) do not widely exist and that tools for heterogeneous platforms depend on significant human intervention and thus, automation does not widely exist. For HSoCs (type 3), on the other hand, more tools are available, and designs including a simple processor plus an accelerator dominate.

We distinguish tools into five different types: languages, application libraries and runtime/OS, compilers, simulation/performance modeling tools, and debuggers. Moreover, we relate the activities of each of these types of tools for software development in typical HECS design flow in six main thrusts listed below:

- Partitioning
  - Partition applications into parts which show affinity towards different heterogeneous components
  - Load balancing

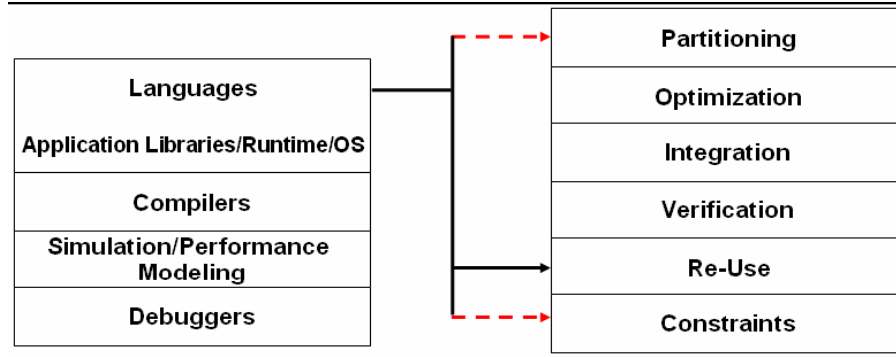


**Figure 24:** Key costs of HECS design flow

- Optimization
  - Optimize each partition
- Integration
  - Tools and abstractions to encapsulate communication/synchronization between components
- Verification
  - Debugging and simulation
- Re-use
  - Commonly required functionality
- Constraints
  - A way of expressing and achieving constraints

The tool activities for languages are shown in Figure 25. Languages help with partitioning by using explicit threading, which in current technologies constitutes low productivity, very difficult, no automatic load balancing. Re-use is typically enabled by language design; for example, “intra-application” re-use is performed via object oriented languages, generics, and templates. Language capability to express constraints, on the other hand, is still in research phase.

As for the application libraries and runtime/OS tools, the current tools for integration, re-use and constraints are reasonably mature, whereas the optimization activity still needs improvement, as optimization is typically performed by hand. In case of compilers, automatic compiler-based partitioning is still under research, and mapping onto FPGAs and course-grained reconfigurable architectures (CGRA) are just beginning to be explored



**Figure 25:** Tools and their activities (this example relates languages to activities)

in the state-of-the-art. While component-specific optimization is well understood (e.g., loop transformations, optimizations for parallelism, machine specific optimizations), very few optimizations are in place for heterogeneous platforms (type 1). Simulation and performance modeling, on the other hand, is typically used for verification: our survey results indicate that simulation time is still an issue, and simulations are performed at several levels. Fast simulation and performance modeling is desired for optimization purposes, however, this cannot to be done in current technologies, as simulation time and re-targeting the simulator are main problems.

In summary, designs at this level are very much based on designer experience, thus ingenuity instead of intelligent tool-based design is essential. We observe that language level research is needed to specify constraints, enable partitioning and optimization; libraries are needed for automatic specialization and optimization—although operating systems are relatively mature; compilers for automatic partitioning (for satisfying constraints and transparently leveraging libraries) are desired; and fast simulation and performance modeling are also desired to enable design space exploration as well as verification of HECS design and use.

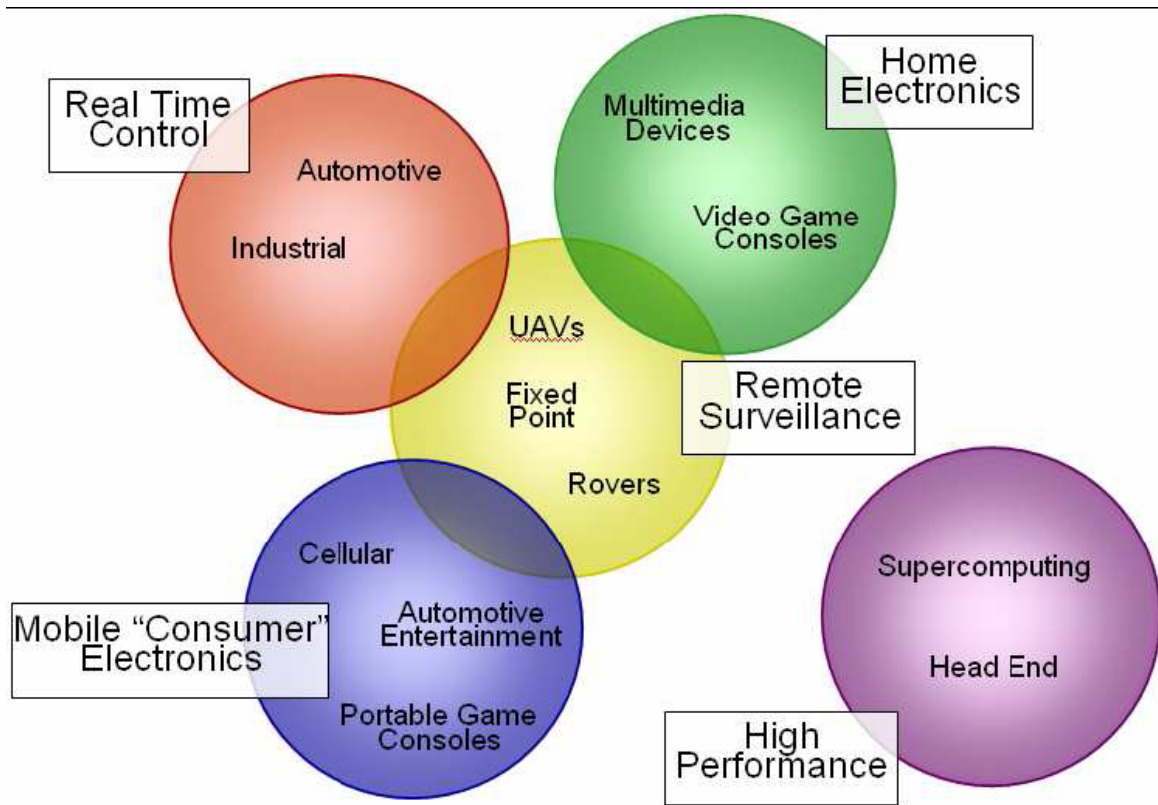
### 3.2.1.3 Applications

In our survey of HECS applications, we distinguished applications into five domains: real-time control applications, remote surveillance, home electronics, mobile consumer electronics, and high performance applications. These five domains are shown in Figure 26. Below, we list the behavior of each domain of applications with examples.

#### 1. Real-time Control Applications:

Behavior:

- Catastrophic Failure
- Safety Critical
- Hard Timing Deadlines
- Real-time Operation
- Operational Degradation
- Components Wear Over Time



**Figure 26:** Application domains

Examples:

- Automotive

Volvo S80

K. Hanninen, J. Maki-Turja, and M. Nolin. Present and future requirements in developing industrial embedded real time systems. In Proc. of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS06), pages 139-150, 2006.

- Industrial

F. Jammes and H. Smit. Service-oriented paradigms in industrial automation. IEEE Transactions on Industrial Informatics, 1(1):62 70, Feb. 2005.

## 2. Remote Surveillance Applications:

Behavior:

- Remote Operation (limited bandwidth, difficult to service)
- Soft Real-Time Deadlines
- Constrained Power
- Constrained Size and Weight (varies by application)

Examples:

- Fixed Point Surveillance

Smart Cam (W. Wolf, Princeton University)

- UAVs

Aersonde (Aerosonde Proprietary Ltd.): small unmanned surveillance plane with >30 hrs. endurance

Carolo T200 (Institute of Aerospace Systems, Technical University of Braunschweig, Germany): micro UAV, takeoff weight 5 kg

- Rovers

Mars Exploration Rovers (JPL): The Spirit and Opportunity rovers

## 3. Home Electronics Applications:

Behavior:

- Soft Timing Deadlines (Real-time Operation)
- Loose Size, Weight, & Power Limitations (only driven by economics of solution)
- Ease of Use Critical

Examples:

- Multimedia Devices

Embedded Software Drive the Digital Home, by C. Hung and R. Schmitt ([www.embedded-computing.com/articles/id/?248](http://www.embedded-computing.com/articles/id/?248))

- Media Connector (Digital Deck)

Video Game Consoles

PlayStation 3 (Sony)

#### 4. Mobile Consumer Electronics Applications:

Behavior:

- High Performance with Extreme Size & Weight Limitations
- Constrained Power

Examples:

- Cellular  
Razr (Motorola)
- Automotive Entertainment
- Portable Game Consoles  
iPod (Apple)

#### 5. High Performance Applications:

Behavior:

- Heat, Size & Power Sensitive  
Data center scale operation limits heat removal, power distribution, and overall size
- Extreme Computational Loading

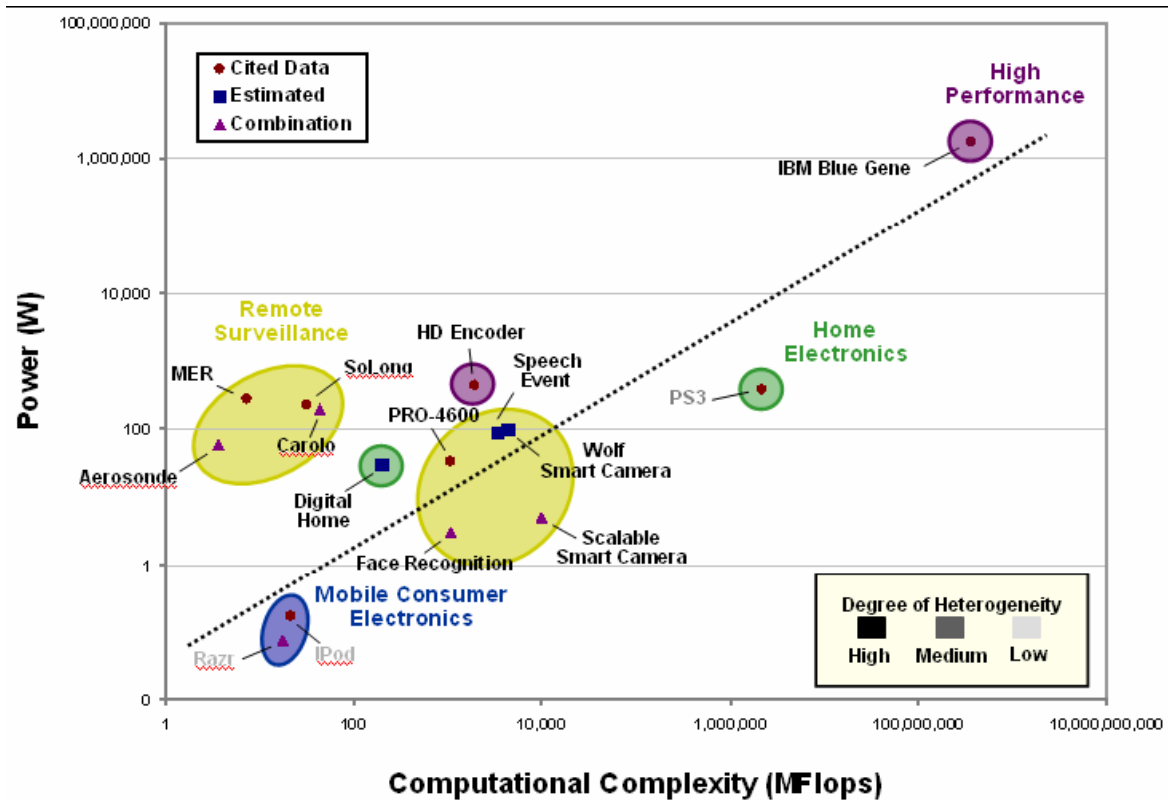
Examples:

- Supercomputing  
IBM Blue Gene/L (IBM): Fundamental science simulation
- Head End  
AVC HD Encoder (Scientific Atlanta): Real-time HD encoder for HDTV broadcast

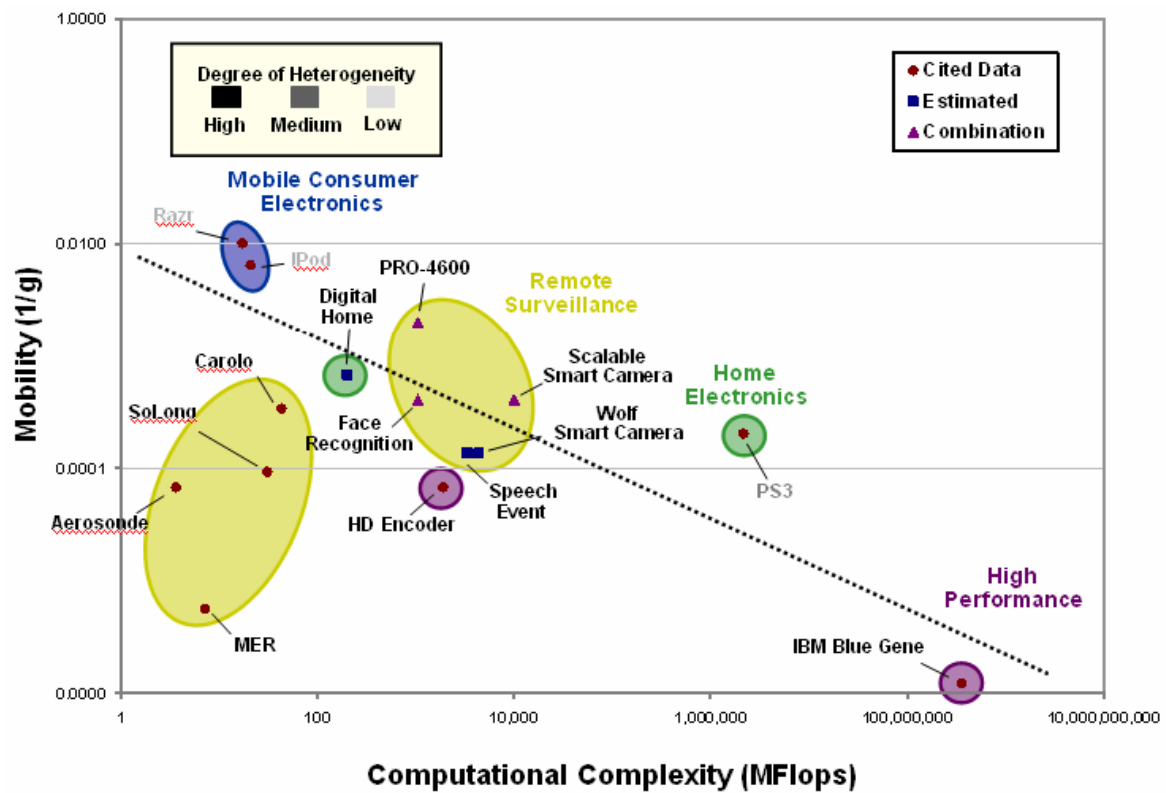
As a result of our survey analysis, we showed that higher computational complexity correlates with higher platform power, and that higher computational complexity inversely correlates with platform mobility. Please see Figures 27 and 28.

Home electronics applications are primarily driven by the economics of the solution. Real time control applications are generally employed on platforms with ample power for electronics. Size is only constrained due to the sheer volume of the electronic systems employed. In remote surveillance, application size is a concern limiting mobility; however, the overall size of the platform requires significant power consumption to provide mobility and allows ample power for electronics. Mobile consumer electronics application platforms require tight size constraints allowing for extremely limited power capacity. Due to the solution scale required to meet the extreme computational demands, high performance benefits greatly from reduced power consumption and size.

As far as application development is concerned, tools are needed for a variety of reasons: Video and audio processing requirements in home electronics typically leads to multiprocessor solutions facilitating the need for tools for code mapping to each processor. Extreme computational loading seen in the high performance class coupled with limited power, heat, and volume budgets necessitates performance analysis tools. Remote operation and audio/video processing requirements seen in remote surveillance drive the need for multiple processors and low power operation creating a need for code mapping and performance analysis. High levels of heterogeneity and critical system availability in real time control combine to make validation both difficult and essential. Finally,



**Figure 27:** Higher computational complexity correlates with higher platform power



**Figure 28:** Higher computational complexity inversely correlates with platform mobility



consumer demands for small form factor solutions with extended battery life in mobile consumer electronics makes performance analysis tools crucial.

### **3.2.2 Productivity challenges and a compiler centric approach**

The productivity challenges stemming from HECS architectures and their implications for tools and automation can be summarized as follows. Design space for HECS architecture platforms is very rich. Attributes for programmable tiles constitute six categories and around twenty different attributes; attributes for configurable tiles constitute five categories and around twenty different attributes; attributes for interconnect constitute three categories and around ten different attributes. As an example, the example platforms from our survey include up to 24 different chips, each including a variety of specialized hardware elements, cores, accelerators, interconnects, etc., which is a huge design space. Under such complexity, there is also a need for meeting (application) constraints (in terms of power, performance, etc). As for our survey examples, power ranges from 50mW to 450W, performance ranges from 0.1GFlops to 400GFlops, power-per-performance ranges from 0.7GFlops/W to 3GFlops/W, interconnect data rate ranges from 1 MB/s to 100 GB/s, plus additional requirements exists due to configurability, flexibility, etc.

Given the aforementioned design complexity and huge design space, the existing technologies offer very limited tools and automation. Problems due to the fact that variety of design flows exist and that integration and development depend on human experience and intervention, verification and integrated hardware and software development become extremely difficult. As a result, design and use of HECS are exposed to high non-recurring engineering (NRE) costs, low productivity, and high risk. Thus, productivity oriented HECS-design flow is needed. Overall, application complexity and architectural richness combine to increase design/implementation effort, thus, reduced productivity in development. We summarize the hurdles of productivity in Figure 29.

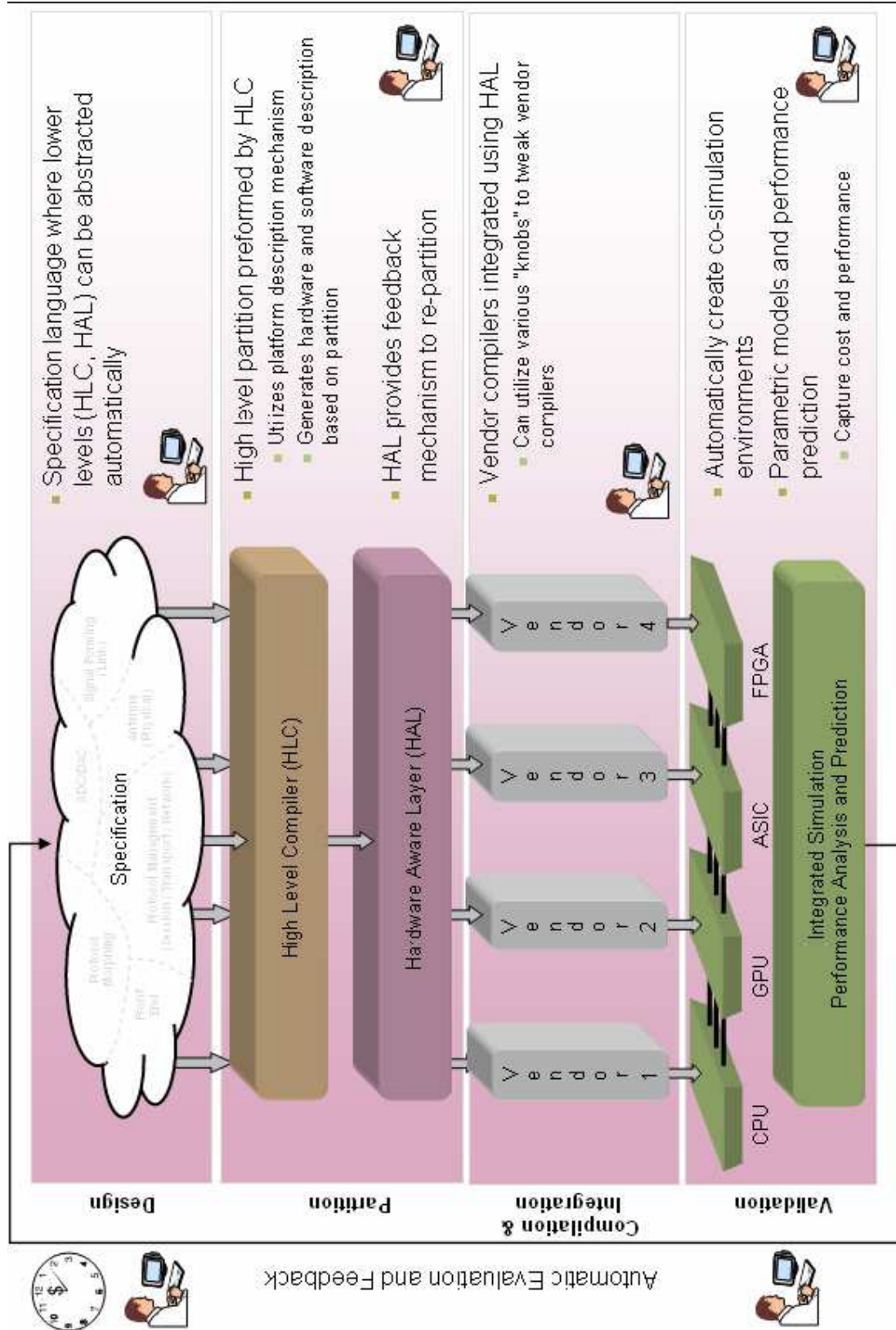
As a solution to aforementioned problems, we proposed a compiler centric approach to massively improved productivity. Our approach is illustrated in Figure 30. Specifically, our approach proposes use of telescoping languages and modeling at the design stage; cognitive adaptation, compiler optimization, and modeling at the partition stage; and again modeling at the compilation and integration stage. Specific list of challenges as well as our proposed approaches to overcome each challenge are shown in Table 3.

## **3.3 Reverse Engineering of Legacy Applications**

To highlight a few, sources for legacy code include porting of software into a new/updated platform or into a new language, web-based Java applications (e.g., unified modeling language (UML) modeling and reverse engineering Java applications) and code transitions among third party vendors (e.g., outsourcing). As far as the languages concerned, the examples are C, C++, Java, C#, C++/CLI, VB, FORTRAN, COBOL, Smalltalk and Lisp. Below, we list some legacy system migration examples (taken from [2]).

- JOVIAL73 on MIL1750 → C on PowerPC  
Military avionics + weapons management





**Figure 30:** A compiler centric approach to massively improved productivity

- COBOL74 + IDMS → COBOL85 + SQL
- UNISYS 1100 retirement; must move data, too!
- K&R C + custom RTOS → ANSI C + VXworks
- Microprocessor modernization
- Clipper + green screen → Delphi + GUI
- Legacy 3GL data processing language
- MODCOMP ASM → C
- Defense radar modernization; 12 computer languages
- Verilog → VHDL
- Reuse of chip design in new context

The main problem statement of dealing with legacy code can be summarized as follows. We are given a large extant code base of very useful code, however, it is not understood by the people currently developing it (e.g., developing new applications and/or working with new platforms). Thus, we need to help innovate tools and technologies to generate well-understood high level versions of the code through reverse engineering. This can be realized through semiautomatic generation of high level code and use of cognitive techniques. We also need innovative tools and technologies to enable new versions of the code to be used by the current set of users and developers without any productivity loss.

As a solution to the aforementioned problems, we proposed a semantic amplification through cognitive techniques. This bridges semantic gap through limited human intervention. Human inserted cues are utilized by compiler along with analysis, such that, the visual representation of code is produced, and programmer intention is preserved. One other important technique of our approach is to use syntax based methods—tree pattern matching based syntax-driven translation methods—so as to translate program to a higher level language (e.g., Matlab). Finally, our approach uses probabilistic modeling and computational learning: example approaches are statistical approaches and neural networks based cognitive techniques for pattern recognition (akin to approaches for natural language translation).

### **3.4 *Software Standards and Techniques for Cognitive and Commodity Computing Systems***

#### **3.4.1 GPU SVM Task**

In accordance with the plan outlined in Chapter 2.4.2, GT developed a library implementation of the SVM for GPU, in coordination with Reservoir Labs. In addition, GT assisted Reservoir with the development of the required LLC. Preliminary findings were published at the High Performance Embedded Computing Workshop at Lincoln Laboratory in September [30], and also reported to the Morphware Forum at a regular Forum meeting Nov. 30 - Dec. 1, 2005.

Subsequent to the initial development and reporting, GT continued to tune and benchmark the SVM-GPU library. In addition, GT participated in outreach activities with the two major vendors of GPUs. These activities were a working group meeting on the topic of “Programmability for General Purpose Computing on GPUs”, hosted by nVidia,

and a meeting with ATI Technology's compiler group to discuss GPU programmability issues. In mid-2006, an interferometric (3D) synthetic aperture radar IFSAR phase unwrapper experiment was added to the GPU tasking as a limited experiment on a code of interest in DoD sensor processing communities. The intent was to develop a GPU-appropriate algorithm and benchmark it in "native" C and CPU (Cg + OpenGL) implementations. This algorithm could then be converted to the GPU SVM API to provide additional testing and evaluation of that API. A specific least-squares algorithm was identified and prototyped in MATLAB. Initial implementations in C on a conventional uniprocessor and on the GPU were developed and benchmarked. GPU speedups of approximately 35x (depending on problems size) were obtained relative to uniprocessor C code. These results were documented in a paper published and presented at the IEEE Radar 2007 Conference in April, 2007 [22].

## CHAPTER 4

### CONCLUSIONS

#### 4.1 *Power Aware Computing Task*

As device scaling continues into nanometer regime, noise is emerging as an increasing threat to reliable computing. In addition, energy consumption and the accompanying heat dissipation also emerge as a limiting factor. In this work, we have introduced PCMOS as a novel technology to overcome both of these hurdles. A thorough characterization of PCMOS behavior and the utilization of noise in a controlled manner enables us to design architectural building blocks. Using such architectures which leverage PCMOS technology, we have demonstrated the impact of PCMOS at the application level. Several applications from the embedded domain, ranging from the domain of security to speech and image processing, are based on probabilistic and error-tolerant algorithms and can readily leverage PCMOS technology. Having established its utility in the context of probabilistic architectures, we also studied impact of voltage overscaling and hence associated propagation delays on probabilistic behavior for arithmetic units (adders and multipliers) and DSP primitives. Our work to this end also constitutes a basis to encompass parameter variations which we envision to be treated as another source of noise and hence probabilistic behavior.

#### 4.2 *HECS Task*

Ever increasing computational power necessitates increases in design productivity to combat the associated rising product development costs. A parameterized, modular HECS architecture platform definition, together with a compiler centric approach helps automate the development cycle for HECS design. By exposing architectural details, through a parametric model, a compiler can perform system level software design alleviating the need for a human to handle the complex task of software partitioning and integration. The resulting improvements in productivity will serve to drive HECS development costs down.

#### 4.3 *Reverse Engineering Task*

One effective approach would be to use human guided cognitive methods to extract an efficient and feasible grammar and associated set of production rules such that a language representation can in turn be raised to a more abstract application specification efficiently. Current art does not support any automation; thus, any form of automation is intended to yield one to two orders of magnitude of improvements to productivity at least.

## REFERENCES

- [1] Akgul, B. E. S., Chakrapani, L. N., Korkmaz, P., and Palem, K. V., “Probabilistic CMOS technology: a survey and future directions,” *Proc. of IFIP International Conference on VLSI SoC*, Oct. 2006.
- [2] Baxter, I. D., “Automated porting of software systems using DMS,” 2001.
- [3] Beinlich, I., Suermondt, G., Chavez, R., and Cooper, G., “The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks,” *Proceedings of the Second European Conference on AI and Medicine*, pp. 247–256, 1989.
- [4] Boltzmann, L., “Further studies on the equilibrium distribution of heat energy among gas molecules,” *Viennese Reports*, Oct. 1872.
- [5] Carnot, S., *Reflections on the Motive Power of Fire (Reflexions sur la Puissance Motrice du Feu et sur les Machines Propres a Developper Cette Puissance)*. 1824.
- [6] Chaitin, G. and Schwartz, J., “A note on monte carlo primality tests and algorithmic information theory,” *Comm. Pure and Applied Math.*, vol. 31, pp. 521–527, 1978.
- [7] Chakrapani, L. N., Akgul, B. E. S., Cheemalavagu, S., Korkmaz, P., Palem, K. V., and Seshasayee, B., “Ultra efficient embedded SOC architectures based on probabilistic cmos technology,” in *Proceedings of the 9th Design Automation and Test in Europe (DATE)*, pp. 1110–1115, Mar. 2006.
- [8] Chakrapani, L. N., Gyllenhaal, J., mei W. Hwu, W., Mahlke, S. A., Palem, K. V., and Rabbah, R. M., *Trimaran: An Infrastructure for Research in Instruction-Level Parallelism*, vol. 3602. Springer-Verlag Berlin Heidelberg, Aug. 2005.
- [9] Chakrapani, L. N., Korkmaz, P., Akgul, B. E. S., and Palem, K. V., “Probabilistic system-on-a-chip architectures,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, Aug. 2007.
- [10] Cheemalavagu, S., Korkmaz, P., and Palem, K. V., “Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship,” in *Proc. of the International Conference on Solid State Devices and Materials*, (Tokyo, Japan), pp. 402 – 403, Sept. 2004.
- [11] Cheemalavagu, S., Korkmaz, P., Palem, K. V., Akgul, B. E. S., and Chakrapani, L. N., “A probabilistic CMOS switch and its realization by exploiting noise,” *Proc. of IFIP International Conference on VLSI SoC*, pp. 452–458, Oct. 2005.

- [12] Clausius, R., “Über die Bewegende Kraft der Wärme,” *Annalen der Physik*, 1850.
- [13] Ding, Y. Z. and Rabin, M. O., “Hyper-encryption and everlasting security,” *Lecture Notes In Computer Science; Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, vol. 2285, pp. 1–26, 2002.
- [14] Fuks, H., “Non-deterministic density classification with diffusive probabilistic Cellular automata,” *Physical Review E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 66, no. 066106, 2002.
- [15] Gelenbe, E., *Neural Networks Advances and Applications*. Netherlands: Elsevier Science Publishers B. V., 1991.
- [16] Gelenbe, E. and Batty, F., “Minimum graph covering with the random neural network model,” in *Neural Networks: Advances and Applications*, vol. 2, 1992.
- [17] George, J., Marr, B., Akgul, B. E. S., and Palem, K. V., “Probabilistic arithmetic and energy efficient embedded signal processing,” in *Proc. Intl. Conf. on Compilers, Architecture and Synthesis for Embedded Systems CASES*, (Seoul, Korea), pp. 158–168, 2006.
- [18] Gibbs, J. W., “On the equilibrium of heterogeneous substances,” *Transactions of the Connecticut Academy*, vol. 2, p. 108248, 1876.
- [19] H. H. Jones, “How to slow the design cost spiral.”
- [20] ITRS 2005 Edition, “<http://www.itrs.net/common/2005itrs/execsum2005.pdf>.”
- [21] J. P. Eckert and J. Mauchly, “Electronic numerical integrator and computer (ENIAC). US Patent No. 3,120,606..”
- [22] Karasev, P. A., Campbell, D. P., and Richards, M. A., “Obtaining a 35x speedup in 2d phase unwrapping using commodity graphics processors,” in *IEEE Radar Conference*, (Boston, MA), pp. 574–578, 2007.
- [23] Korkmaz, P., Akgul, B. E. S., Palem, K. V., and Chakrapani, L. N., “Advocating noise as an agent for ultra-low energy computing: Probabilistic CMOS devices and their characteristics,” *Japanese Journal of Applied Physics, SSDM Special Issue Part I*, pp. 3307–3316, Apr. 2006.
- [24] Korkmaz, P. and Palem, K. V., “The inverse error function and its asymptotic “order” of growth using O and !,” *Tech. Rep. CREST-TR-06-02-01*, Georgia Institute of Technology, GA, Feb. 2006.



- [25] Korkmaz, P., Akgul, B. E. S., Chakrapani, L. N., and Palem, K. V., “Advocating noise as an agent for ultra low-energy computing: Probabilistic CMOS devices and their characteristics,” *Japanese Journal of Applied Physics (JJAP)*, vol. 45, pp. 3307–3316, Apr. 2006.
- [26] Korkmaz, P., Akgul, B. E. S., and Palem, K. V., “Ultra-low energy computing with noise: energy-performance-probability trade-offs,” *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 349–354, Mar. 2006.
- [27] Korkmaz, P., Akgul, B. E. S., and Palem, K. V., “Analysis of probability and energy of nanometre cmos circuits in presence of noise,” *Electronics Letters*, vol. 43, pp. 942–943, Aug. 2007.
- [28] Leff, H. and Rex, A., eds., *Maxwell’s Demon: Entropy, Information, Computing*. Princeton University Press, 1990.
- [29] Lyonnard, D., Yoo, S., Baghdadi, A., and Jerraya, A. A., “Automatic generation of application-specific architectures for heterogeneous multiprocessor system-on-chip,” *Proc. of Design Automation Conference*, pp. 518–523, 2001.
- [30] Mackenzie, K., Campbell, D., and Szilyagi, P., “A streaming virtual machine for GPUs,” in *9th High Performance Embedded Computing Workshop*, (Lexington, MA), September 2005.
- [31] Meindl, J. D. and Davis, J. A., “The fundamental limit on binary switching energy for terascale integration (TSI),” *IEEE Journal of Solid State Circuits*, vol. 35, pp. 1515–1516, Oct. 2000.
- [32] Palem, K. V., “Proof as experiment: Probabilistic algorithms from a thermodynamic perspective,” in *Proc. Intl. Symposium on Verification (Theory and Practice)*, (Taormina, Sicily), June 2003.
- [33] Palem, K. V., “Energy aware computing through probabilistic switching: A study of limits,” *IEEE Trans. Computer*, vol. 54, no. 9, pp. 1123–1137, 2005.
- [34] Palem, K. V., Akgul, B. E. S., and George, J., “Variable scaling for computing elements,” *Invention Disclosure*, Georgia Tech, Feb. 2006.
- [35] Palem, K. V., Cheemalavagu, S., Korkmaz, P., and Akgul, B. E., “Probabilistic and introverted switching to conserve energy in a digital system,” *US Patent*, no. 20050240787, 2005.
- [36] Park, S. and Miller, K. W., “Random number generators: good ones are hard to find,” *Communications of the ACM*, vol. 31, Oct. 1988.

- [37] Pavlovic, V., Garg, A., Rehg, J. M., and Huang, T. S., "Multimodal speaker detection using error feedback dynamic bayesian networks," in *Computer Vision and Pattern Recognition*, vol. 2, (Ft. Collins, CO), pp. 34–41, June 2000.
- [38] Rabin, M. O., "Probabilistic algorithms," in *Algorithms and Complexity, New Directions and Recent Trends* (Traub, J. F., ed.), pp. 29–39, 1976.
- [39] Rehg, J. M., Murphy, K. P., and Fieguth, P. W., "Vision-based speaker detection using bayesian networks," in *Computer Vision and Pattern Recognition*, (Ft. Collins, CO), pp. 110–116, June 1999.
- [40] Schwartz, J. T., "Fast probabilistic algorithms for verification of polynomial identities," *Journal of ACM*, vol. 27, pp. 701–717, Oct. 1980.
- [41] Sinha, A. and Chandrakasan, A., "JouleTrack a web based tool for software energy profiling," *Proc. of Design Automation Conference*, pp. 220–225, 2001.
- [42] SMIPS32 M4K Processor Core Software Users Manual, "<http://www.mips.com>."
- [43] Stein, K.-U., "Noise-induced error rate as a limiting factor for energy per operation in digital ICs," *IEEE J. Solid-State Circuits*, vol. 12, pp. 527–530, Oct. 1977.
- [44] Strecok, A. J., "On the calculation of the inverse of the error function, mathematics of computation," *Mathematics of Computation*, vol. 22, pp. 144–158, Jan. 1968.
- [45] StrongARM-1100 Microprocessor Technical Reference Manual, "<http://www.intel.com>."
- [46] Trimaran Consortium, "Trimaran: An infrastructure for research in instruction-level parallelism."
- [47] von Neumann, J., "First draft of a report on the EDVAC," Contract No. W-670-ORD-492, Moore School of Electrical Engineering, Univ. of Penn., Philadelphia, 30 June 1945.
- [48] Wang, J.-C., Wang, J.-F., Suen, A.-N., and Weng, Y.-S., "A programmable application-specific VLSI architecture for speech recognition," The 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS), vol. 1, pp. 477–480, 2001.
- [49] Xtensa Microprocessor, "<http://www.tensilica.com>."

## LIST OF ACRONYMS

ACIP - Architectures for Cognitive Information Processing  
AMI - American Microsystems Incorporated  
API - Application Programming Interface  
ASIC - Application-Specific Integrated Circuit  
BIVOS - Biased Voltage Scaling  
BN - Bayesian Network  
CBEA - Cell Broadband Engine Architecture  
CGRA - Course-Grained Reconfigurable Architectures  
CMOS - Complementary Metal Oxide Semiconductor  
DSP - Digital Signal Processing  
EPP - Energy-Performance Product  
EDP - Energy-Delay Product  
EIB - Element Interconnect Bus  
FFT - Fast Fourier Transform  
FIR - Finite Impulse Response  
FPGA - Field Programmable Gate Array  
GT - Georgia Institute of Technology  
GPU - Graphical Processing Units  
HAL - Hardware Aware Layer  
HD - High Definition  
HDL - Hardware Description Language  
HE - Hyper Encryption  
HECS - Heterogeneous Embedded Computing System  
HLC - Higher Level Compiler  
HSoC - Heterogeneous System on a Chip  
IFSAR - Interferometric (3D) Synthetic Aperture Radar  
ITRS - International Technology Roadmap for Semiconductors  
LFF - Living Framework Forum  
LLC - Low Level Compiler  
MIPS - Microprocessor without Interlocked Pipeline Stages  
MSI - Morphable Stable Interface  
NRE - Non-Recurring Engineering  
NSR - Noise to Signal Ratio  
OS - Operating System  
PBITs - Probabilistic bits  
PCA - Polymorphous Computing Architectures  
PbCA - Probabilistic Cellular Automata  
PCMOS - Probabilistic CMOS  
PPE - Power Processor Element  
PRNG - Pseudo-Random Number Generator  
PSOC - Probabilistic System-On-a-Chip  
RISC - Reduced Instruction Set Computer

RNN - Random Neural Networks  
SAR - Synthetic Aperture Radar  
SNR - Signal to Noise Ratio  
SPE - Synergistic Processing Elements  
SVM - Stream Virtual Machine  
TSMC - Taiwan Semiconductor Manufacturing Company  
UML - Unified Modeling Language  
VHDL - Very-high-speed integrated circuits Hardware Description Language  
VM - Virtual Machine